Weighted Model Procedure For Federated Learning (WgtFed)

Prajjwal Singh , Pon Harshavardhanan

October 2023

Abstract

A distributed machine learning technique called federated learning allows model training on a vast corpus of decentralized data. With federated learning, thousands of users can build a deep learning model without disclosing to one another their personal training data.

In this paper, Design a federated learning system to shorten convergence time by incorporating weighted federated averaging, which will determine the quantity of data to be analyzed through dynamic scheduling. Design the federated computation to generalize the federated learning using Map-Reduce (Master-Slave architecture) instead of TensorFlow. we will use dynamic federated averaging or weighted federated averaging, we propose a new method in this paper that will improve efficiency and communication over federated averaging algorithms. An extensive empirical evaluation validates major improvement of the trade-off between model performance and communication which could be beneficial for numerous decentralized learning applications, such as autonomous driving, or voice recognition and image classification on mobile phones.For instance, several cellphones working together can train a keyboard’s next-word predictor without disclosing what specific users input.

1 Introduction

As datasets grow larger and models more complex, training machine learning models increasingly requires distributing the optimization of model parameters over multiple machines. Existing machine learning algorithms are designed for highly controlled environments (such as data centers) where the data is distributed among machines in a balanced and i.i.d. fashion, and high-throughput networks are available.

Federated Learning (FL) is a distributed machine learning approach which enables training on a large corpus of decentralized data residing on devices like mobile phones. FL is one instance of the more general approach of “bringing the code to the data, instead of the data to the code” and addresses the fundamental problems of privacy, ownership, and locality of data.

In federated learning, a large number of clients with erratic and comparatively sluggish network connections share training data with one another in an effort to create a high-quality centralized model. We take into consideration learning techniques in this scenario, where each client individually computes an update to the current model based on its local data and sends this update to a central server, where the client-side updates are combined to compute a new global model. In this context, mobile phones are the typical clients, thus effective communication is crucial.

When training data is derived from users' interactions with mobile applications, this presents a compelling case for federated learning. Federated Learning removes the requirement for cloud storage of training data, allowing mobile phones to jointly develop a shared prediction model while retaining all training data locally. Mobile devices owned by users store the training data locally, and these devices function as nodes, processing and updating a global model using their local data. This extends the use of local models to mobile devices by introducing model training directly into the device.

If a decentralized learning protocol is adaptive and consistent, it is considered efficient. Every one of the requirements is easily met: While a protocol that centralizes all data is consistent but not adaptive, one that does not synchronize is adaptive but not consistent. Periodically communicating protocols are consistent, meaning they attain a prediction performance that is on par with a centrally learned model on all available data. Nevertheless, regardless of the loss, they demand a level of communication that is linear in the number of learners (m) and the number of rounds (T). They are therefore not adaptable.

Decoupling model training from the requirement for direct access to the raw training data is one of this method's main advantages. Evidently, there is still a need for some degree of trust in the server managing the training, and the updates can still contain private information based on the specifics of the model and algorithm.

Federated learning, on the other hand, can drastically lower privacy and security concerns for applications where the training aim can be determined based on data accessible on each client by restricting the attack surface to just the device rather than the device plus the cloud.

Choosing between synchronous and asynchronous training algorithms is a fundamental design choice for a Federated Learning system. Even in the data center, there has been a steady tendency recently toward synchronous big batch training, despite the fact that much of the effective work on deep learning has employed asynchronous training.

When compared to periodically communicating state-of-the-art techniques, this results in an order of magnitude reduction in communication. In addition, we obtain a communication bound that varies well with the serialized learning problem's difficulty. Since the predictive performance doesn't really change, there is almost no cost associated with the communication reduction.

In a similar vein, the weighted Federated Averaging algorithm operates. Additionally, a number of methods for improving privacy guarantees for FL—such as differential privacy—basically need for some kind of device synchronization so that the learning algorithm on the server side only has to ingest a basic aggregate of the updates from several users. Considering all of these factors, we decided to concentrate on synchronous round support while reducing any possible synchronization cost using a number of methods that we will go over in more detail later.

In-fleet learning of autonomous driving functionalities is a natural application domain for dynamic decentralized machine learning: concept drifts naturally arise from properties central to the modeling task changing traffic behavior over time and from constant and unpredictable concept drifts introduced by different countries or regions.

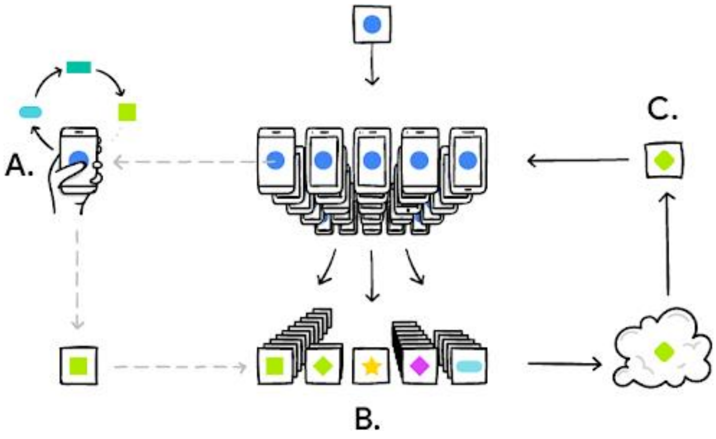

Figure 1: Federated Learning

Furthermore, in big fleets, data centralization becomes prohibitively expensive due to the huge high-frequency data streams produced by several sensors per vehicle. In-fleet learning of autonomous driving functions is one of the synthetic and real-world challenges with concept drift on which this study will present a thorough empirical evaluation of the dynamic averaging technique.

2 Literature Review

Contributions of the paper ”towards federated learning at scale: system design” by Keith Bonawitz (2018) include:

Building a scalable production system for Federated Learning on mobile devices using TensorFlow. Describing the high-level design of the system, addressing challenges, and providing solutions. Identifying open problems and future directions for Federated Learning. Focusing on support for synchronous rounds and mitigating synchronization overhead through various techniques.

Providing a system design for Federated Learning algorithms in the domain of mobile phones (Android). Presenting operational profile data for one of the Federated Learning populations and discussing the Finalization phase of the protocol. Exploring the potential applications of the general deviceserver architecture beyond machine learning, such as Federated Computation and Federated Analytics. Acknowledging the contributions of multiple individuals to the design and implementation of the system. Contributing to the field of systems research by outlining the major components of the system, describing challenges, and identifying open issues.

In a similar vein, the weighted Federated Averaging algorithm operates. Additionally, a number of methods for improving privacy guarantees for FL—such as differential privacy—basically need for some kind of device synchronization so that the learning algorithm on the server side only has to ingest a basic aggregate of the updates from several users. Considering all of these factors, we decided to concentrate on synchronous round support while reducing any possible synchronization cost using a number of methods that we will go over in more detail later.

In-fleet learning of autonomous driving functionalities is a natural application domain for dynamic decentralized machine learning: concept drifts naturally arise from properties central to the modeling task changing traffic behavior over time and from constant and unpredictable concept drifts introduced by different countries or regions.

Furthermore, in big fleets, data centralization becomes prohibitively expensive due to the huge high-frequency data streams produced by several sensors per vehicle. In-fleet learning of autonomous driving functions is one of the synthetic and real-world challenges with concept drift on which this study will present a thorough empirical evaluation of the dynamic averaging technique.

3 System Design

3.1 Basic Notions

The participants involved in the protocol encompass a range of devices, with a current focus on Android phones, as well as the FL server, which operates as a cloud-based distributed service. These devices, serving as active agents, proceed to inform the server of their readiness to engage in an FL task for a particular FL population. It is worth noting that an FL population is delineated by a globally unique name, which serves as a distinctive identifier for the learning problem or application that is under consideration. Consequently, this name plays a pivotal role in effectively demarcating the specific context in which the FL task is being conducted. Within the realm of FL, an FL task assumes the form of a precise computation that is designated for a given FL population. Such computations may encompass a variety of activities, including training operations executed with pre-defined hyperparameters or the evaluation of trained models based on the localized data available on the respective device. Ultimately, the FL task serves as a critical instrument in fostering the advancement and refinement of machine learning techniques.

During a specific time window, where there is a potential for tens of thousands of devices to announce their availability to the server, the server goes through a selection process to choose a subset of devices that will be invited to participate in a particular Federated Learning (FL) task. This process, which we refer to as a ”rendezvous” between the devices and the server, is essentially a round of selection. Once the devices are selected, they remain connected to the server for the entire duration of the round.

Once the selection process is complete, the server proceeds to inform the chosen devices about the specific computation they need to run. This information is conveyed through an FL plan, which is a data structure that includes a TensorFlow graph and instructions on how to execute it. After establishing the round, the server then sends the current global model parameters and any other necessary state to each participant in the form of an FL checkpoint. This checkpoint essentially represents the serialized state of a TensorFlow session.

Each participant in the round then performs a local computation based on the received global state and its own local dataset. Once the local computation is completed, the participant sends back an update to the server in the form of an FL checkpoint. The server then incorporates these updates into its global state, thus updating the overall progress of the FL task. This process of sending updates, incorporating them into the global state, and repeating the round continues until the desired outcome is achieved.

3.2 Phases

The communication protocol serves as a crucial mechanism that facilitates the progression and evolution of devices in order to further enhance the overarching global, singleton model of a Federated Learning (FL) population during the various rounds that transpire throughout the FL process. These rounds, which are composed of distinct and interconnected phases, play a pivotal role in ensuring the seamless and synchronized exchange of information and updates between the participating devices, thereby fostering a collaborative and cohesive environment for the FL population to thrive and flourish.

Selection The selection process occurs periodically, where devices that meet the eligibility criteria connect to the server by establishing a bidirectional stream. This stream serves the purpose of monitoring the devices’ functionality and coordinating multi-step communication. The server then chooses a subset of connected devices based on specific objectives, such as the optimal number of participating devices (usually a few hundred devices take part in each round). In the event that a device is not selected for participation, the server provides instructions for reconnecting at a later time.

Configuration The server’s configuration is determined by the chosen aggregation mechanism (e.g., simple or Secure Aggregation) for the selected devices. Subsequently, the server transmits the Federated Learning (FL) plan and an FL checkpoint containing the global model to each individual device.

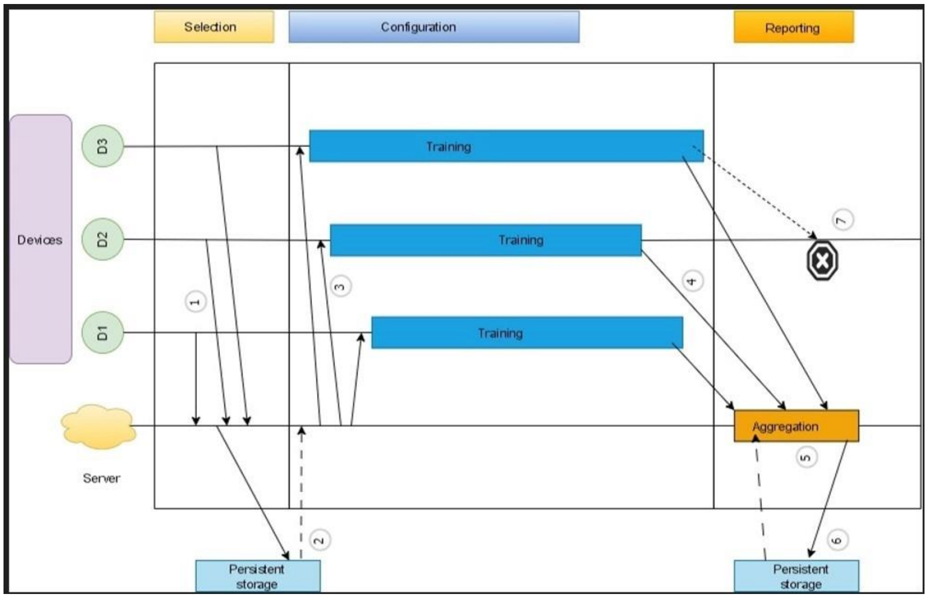

Figure 2: Weighted Federated Learning

Reporting The server waits for the participating devices to report updates. As updates are received, the server aggregates them using weighted Federated Averaging and instructs the reporting devices when to analysis the data as per the amount of data on the device to analyze. When devices report in time, the round will be successfully completed, and the server will update its global model.

The utilization of a communication protocol allows for the progression of devices to enhance the global, singular model of a Federated Learning population between rounds. Each round consists of three distinct phases, as illustrated in figure 2.

(1) Devices initiate a check-in process with the Federated Learning server. If a device is rejected, it is informed to return at a later time.

(2) The server retrieves the model checkpoint from persistent storage.

(3) The model and its corresponding configuration are transmitted to specific devices.

(4) On-device training is executed, and the updated model is subsequently reported back.

(5) As the updates arrive, the server amalgamates them into the global model.

(6) The server then writes the global model checkpoint into persistent storage.

The selection and reporting phases are specified by a set of parameters which spawn flexible time windows. For example, for the selection phase the server considers a device participant goal count, a timeout, a minimal percentage and amount data for dynamic averaging of the goal count which is required to run the round. The selection phase lasts until the goal count is reached.

4 Weighted federated averaging or dynamic averaging

The primary technical achievement of this analysis is the development, execution, and assessment of a flexible mechanism for combining parameters in Federated Learning (FL) referred to as WgtFed. WgtFed introduces multiple groundbreaking aspects:

- WgtFed minimizes the need for state in aggregators and treats aggregators as serverless functions. In many existing FL tasks, each instance of an aggregator processes a sequence of inputs and generates a single output. Any state that exists is not specific to a particular aggregator instance and may be shared among all aggregators. It is more efficient to store such state in an external repository, allowing aggregators to be stateless and, consequently, serverless.

As a result, WgtFed is scalable in terms of the number of participants, making it suitable for cross-silo and cross-device deployments, as well as in terms of geographical distribution, whether it be single/hybrid cloud or multicloud.

- WgtFed leverages serverless technologies to dynamically deploy and remove aggregator instances in response to updates from participating models. This capability enables WgtFed to effectively handle both intermittent and active participants. It is unnecessary for aggregators to be continuously deployed and simply wait for input.

- WgtFed is efficient in terms of resource utilization, thanks to its support for automatic elastic scaling. It also exhibits low aggregation latency.

- WgtFed provides a reasonable level of expressiveness for programmers to easily implement scalable aggregation algorithms.

WgtFed is implemented on the widely used Ray distributed computing platform . It allows the execution of arbitrary Python code in aggregation functions and can utilize GPU accelerators if necessary.

Furthermore, WgtFed enhances the reliability and fault tolerance of FL tasks by reducing the need for state in aggregators, eliminating persistent network connections between aggregators, and employing dynamic load balancing of participants.

4.1 Algorithm

In this section, we show the Federated Averaging algorithm from McMahan et al. (2017) mentioned in algorithm 1 which is taken from the base paper. Broadly speaking, the performance can be evaluated using any metric that captures the effectiveness of the model's proposed solution for the given task, such as accuracy for classification tasks.

Algorithm 1 Federated Averaging targeting updates from K clients per round.

```
SERVER'S UPDATE:
Initialize w0
for each round t = 1,2,... do
    Select eligible clients to compute updates
    ClientUpdate(w) from client k $\varepsilon$ [K].
    Sum of weighted updates
    Sum of weights
    Average update
    $wt + 1 \leftarrow wt + \Delta$
end for
CLIENT'S UPDATE(w)
$\beta \leftarrow$ (local data divided into minibatches)
$n \leftarrow |\beta|$                                  ▷ Update weight
$W init \leftarrow w$
for batch $\varepsilon$ $\beta$ do
    $w \leftarrow wt - \eta\nabla l$
end for
$\Delta \leftarrow n \times (w-W init)$                 ▷ Updated weight
return $(\Delta, n)$ to server
```

Another Algorithm for Weight-Based Federated Learning algorithm is mentioned in Algorithim 2 from A. Giuseppi(2021) Algorithm 2 is reported using a similar syntax to the one used for FedAvg.

WgtFed assesses the federation's clients' contributions to determine their relative weights for the model averaging process. The primary benefits of utilizing WgtFed include prioritizing higher-performing clients to mitigate the detrimental impact of compromised and malicious clients. Additionally, it places increased emphasis on data samples and features that enhance model performance, particularly in scenarios involving rare or challenging labels and characteristics in classification tasks.

WgtFed endeavors to assess the degree of contribution from diverse clients in order to ascertain their respective weights in the model averaging process. In order to accomplish this, WgtFed presupposes the existence of a representative server test set, which enables the evaluation of the performance of the clients' models for the specific task at hand, without causing any substantial escalation in the complexity of the training procedure.

Algorithm 2 WgtFed: Weight-Based Federated Learning algorithm

```
SERVER'S UPDATE:
for each round t = 1,2,...R do
    Select eligible clients to compute updates
    ClientUpdate
    for each client i = 1,2,...K do
        receive the client's model $W t$
        evaluate $Wt$ on the server test set
        use the evaluation to determine the weight $\rho i$
    end for
    update the server's model
end for
CLIENT'S UPDATE(w)
for each client i = 1,2,...K do
    $Wi \leftarrow Ws$
    for each local epoch j = 1, 2, ..., E do
        for each mini-batch b of size B do
            $Wi \leftarrow Wi - n\nabla l(w,b)$
        end for
    end for
    return $Wi$ to server
end for
```

Weighted Model Averaging: When the server executes the model averaging procedure and collects the models submitted by the clients, an initial assessment of all the models is performed on a dataset accessible to the server. Based on the models' performances, different weights are assigned to each model in order to conduct the model averaging process, thereby placing greater emphasis on the better performing models.

5 Performance metrices

We are taken the performance metrices from A. Giuseppi(2021) paper for comparison. In this particular section, the performance of WgtFed is examined and contrasted with that of FedAvg in various scenarios. The underlying objective behind all of the conducted tests is to assess the effectiveness of WgtFed in challenging environments, with the aim of better capturing the impact of its innovative characteristics. Consequently, a comparison will be made between WgtFed and FedAvg, given that FedAvg serves as the standard for federated learning solutions. Throughout the remainder of this section, in order to enhance the clarity of the presentation, it will be assumed that all clients transmit their models to the server during each round of communication.

The initial simulation under consideration pertains to the MNIST dataset. The utilization of such a straightforward dataset facilitates a more comprehensive evaluation of the impact of each of the proposed improvements individually,

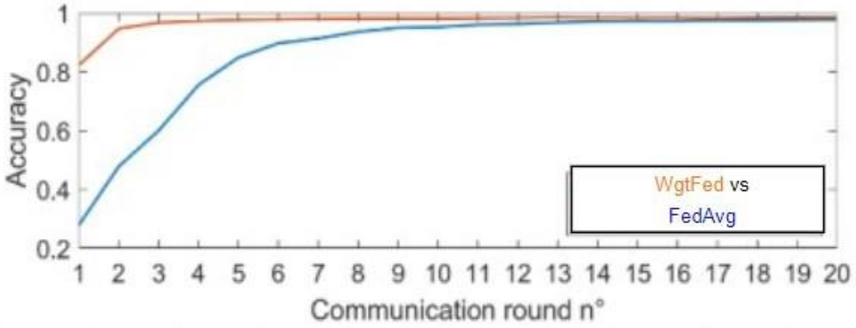

Figure 3: Simulation

in the event that WgtFed is implemented in a specific and temporary manner. We would like to emphasize that the MNIST dataset consists of a collection of 60k+10k labeled images depicting handwritten digits ranging from 0 to 9, and serves as one of the most prevalent benchmarks for classification tasks. The simulation will primarily concentrate on the concept of Weighted Model Averaging.

6 Conclusion and Future Works

In this paper, introduced WgtFed, an innovative Federated Learning technique that utilizes a procedure for weighted model averaging to take into account the varying Weights levels achieved by the clients in the federation. WgtFed enables the handling of non-IID, imbalanced, and highly distributed data, even when confronted with malicious or poorly performing members of the federation. Numerous validation examples were employed to demonstrate that WgtFed achieves commendable performance in both classification and regression tasks, particularly in challenging scenarios. Future research endeavors encompass the incorporation of privacy-aware features into WgtFed and the exploration of its decentralization.

Using Federated Computation I believe there are more applications besides ML for the general device/server architecture or MapReduce technique This is also apparent from the fact that this paper contains no explicit mentioning of any ML logic. Instead, we refer abstractly to ’plans’, ’models’, ’updates’ and so on.

References

[1] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecnˇ y, S. Mazzocchi, H. B. McMahan ‘ et

al., “Towards federated learning at scale: System design,” arXiv preprint arXiv:1902.01046, 2019.

[2] Kamp, M., Adilova, L., Sicking, J., Huger, F., Schlicht, P., Wirtz, T., and Wrobel, S. Efficient decentralized deep learning by dynamic model averaging. arXiv preprint arXiv:1807.03210, 2018.

[3] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016b. Federated Learning of Deep Networks using Model Averaging. arXiv:1602.05629

[4] Konecn ˇ y, J., McMahan, H. B., Yu, F. X., Richt ´ arik, P., ´ Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492, 2016b.

[5] McMahan, H. B. and Ramage, D. Federated learning: Collaborative machine learning without centralized training data, April 2017. URL https://ai.googleblog.com/2017/04/federated-learning collaborative.html. Google AI Blog.

[6] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pp. 1273–1282, 2017.

[7] Konecn ˇ y, J., McMahan, H. B., Ramage, D., and Richt ´ arik, P. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527, 2016a.

[8] Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. arXiv preprint 1811.03604, 2018.

[9] A. Giuseppi, L. D. Torre, D. Menegatti, et al., “AdaFed: Performance-based adaptive federated learning,” in Proc. the 5th International Conference on Advances in Artificial Intelligence, 2021, pp. 38-43.

[10] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konecnˇ y,´ S. Kumar, and H. B. McMahan, “Adaptive federated optimization,” 2020. [Online]. Available: https://arxiv.org/abs/2003.00295