Poster: Maintaining Training Efficiency and Accuracy for Edge-assisted Online Federated Learning with ABS

Jiayu Wang*, Zehua Guo*◇, Sen Liu†, Yuanqing Xia*
*Beijing Institute of Technology †Fudan University

***Abstract*—This paper proposes Adaptive Batch Sizing (ABS) for online federated learning. ABS is an iteration process-efficient solution that adaptively adjusts batch size of the training process at edge nodes. Preliminary results show that ABS maintains training efficiency and accuracy, compared with existing iteration round-efficient solutions.**

I. Introduction

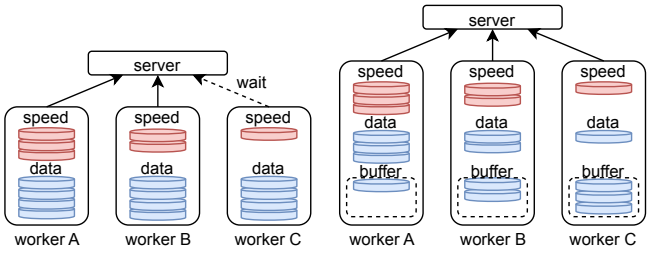
Federated Learning (FL) is a machine learning paradigm that employs distributed devices to assist machine learning training tasks. It outperforms cloud-based centralized machine learning by saving precious bandwidth resource among user devices and the cloud and protecting user privacy [1]. Parameter Server (PS) [2], which consists of workers and servers, is a popular structure for FL. During individual training iteration, each worker retrieves its collected training data, trains machine learning model using parameters received from servers, and returns its updated gradients to servers. Each server updates the global model based on the gradients received from different workers and forwards the updated model to workers for next round training. In FL, workers are usually distributed edge nodes while servers reside in the cloud.

Existing FL solutions (e.g., FederatedAveraging [1]) are usually iteration round-efficient solutions, which aim to reduce the round of training iterations by forcing each worker to use all data at each training iteration. These FL studies implicitly assume that the training data is collected offline, and each worker uses its assigned data for its training. However, in real world, the data are generated and collected from user devices in a real-time manner, and the volume and distribution of data in each worker exhibit temporal and spatial fluctuations [3]. We call the FL using real-time generated data the *online federated learning*. For online federated learning, existing solutions suffer from two issues: (1) the fluctuation of training data batch size led by the varying data receiving rate; (2) heterogeneous iteration processing time among workers, resulting in low utilization of the computing resource.

In this paper, we propose Adaptive Batch Sizing (ABS), an iteration process-efficient solution for online federated learning to achieve high training efficiency and fast convergence under severe data fluctuations. Based on our observation, ABS chooses the training data size of each worker adaptively considering three key factors: processing rate, training model, and training data generation speed. Additionally, ABS also stores or retrieves training data from a training data buffer in each worker based on network conditions. Thus, the fluctuation of the training batch size can be avoided, and the training process is accelerated.

◇This work was supported by National Key Research and Development Program of China under Grant 2018YFB1003700 and Beijing Institute of Technology Research Fund Program for Young Scholars. (Corresponding author: Zehua Guo)

978-1-7281-6992-7/20/$31.00 ©2020 IEEE



(a) Iteration round-efficient solution (b) Iteration process-efficient solution

Figure 1. Comparison of iteration round-efficient solution and iteration process-efficient solution. In the iteration round-efficient solution, waiting for the slow worker's parameter transmission slows down the whole training process. Iteration process-efficient solution stores part of the data in the buffer, and the training process does not decrease the process efficiency.

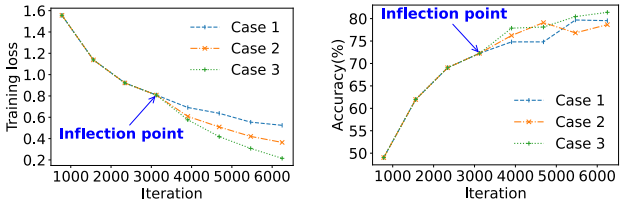
Figure 1 uses an example of an individual training iteration to illustrate the differences between existing iteration round-efficient solutions and ABS in online federated learning. In the example, the FL consists of three workers and one server. Each worker receives 4 normalized units of training data, and the normalized data processing rate of workers A, B, and C are 3, 2, and 1 units per iteration, respectively. In Figure 1(a), worker C processes more slowly than the other two workers, and the server has to wait for the result of worker C before updating its global model. In Figure 1(b), ABS considers the processing rate to configure the batch size of workers A, B, and C to 3, 2, and 1, respectively. Thus, three workers can complete training at the same time, and the training efficiency is improved.

Preliminary simulation results show that ABS outperforms existing iteration round-efficient solutions. Under the same number of training iterations, ABS achieves good training accuracy and reduces training loss by about 20%; ABS accelerates the training process by about 30% when reaching the similar training accuracy and loss.

II. System Design

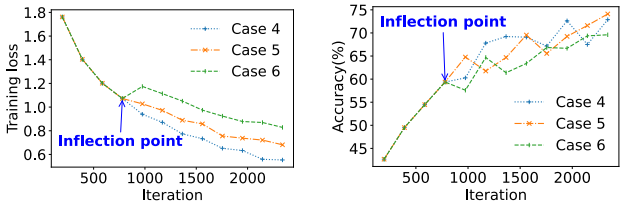
A. Observation

Training batch sizing is an important factor for training process. We conducted several experiments using CNN and CIFAR10 to evaluate the impact of different batch sizes on training loss and accuracy. Figure 2 shows the results. In Figure 2(a) and (b), increasing the batch size can accelerate



(a) Training loss in the case of increased batch size.

(b) Accuracy in the case of increased batch size.



(c) Training loss in the case of decreased batch size.

(d) Accuracy in the case of decreased batch size.

Figure 2. Impact of training batch sizing on training loss and accuracy. Case 1 uses a fixed batch size of 32, and Cases 2 and 3 respectively increase the batch size from 32 to 64 and 128 at the inflection point. In Case 4, we maintain the batch size at 128, and in Cases 5 and 6, we decrease the batch size from 128 to 64 and 32, respectively, at the inflection point.

the training process. In contrast, Figure 2(c) and (d) show that the decrease of the batch size can slow down the training. Note that the extreme small batch size will have a serious negative effect and lead to a long training process duration.

B. System Design

Figure 3 shows the structure of ABS, which includes two phases: warm-up phase and processing phase. The warm-up phase includes the component of **Batch size bound decision**. The component decides the upper and lower bound of the batch size based on warm-up experiments. The server sets a default iteration duration and then decides the lower and upper bound of batch size in this duration. The upper bound ensures that the maximum iteration duration can fall in a certain range. Specifically, we set an iteration duration at first. In each worker, the maximum batch size, which can be processed within this duration, will be set as the upper bound. We set the lower bound to maintain high utilization of the training data. We first collect a certain amount of training data and then train the model with different batch sizes on the training data within one iteration. The batch size with the best training result will be set as the lower bound.

The processing phase consists of three components. **(1) Training data selection:** To prevent frequent batch size fluctuations, workers choose only $C\%$ of the data in the buffer during each iteration. A smaller $C\%$ results in shorter iteration duration while a large C reduces iteration rounds. **(2) Batch size selection:** This component ensures the size of selected data falls in the range of lower bound and upper bound, which are determined during the warm-up phase. **(3) Batch size bound update:** To avoid the training performance degradation resulted from data fluctuation, workers check the batch size in the past K iterations. If the batch size is larger than the lower bound consecutively, the lower bound should be increased at

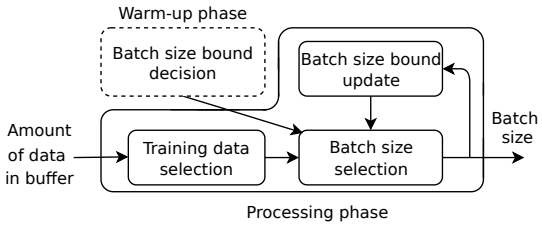
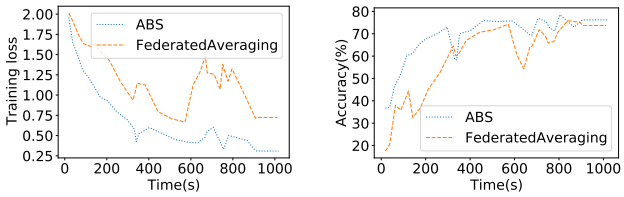


Figure 3. ABS structure.



(a) Training loss.

(b) Accuracy.

Figure 4. Simulation results.

the end of this iteration. A small K will cause the upper and lower bound to overlap rapidly and reduce the source utilization while a large K will fail to alleviate the fluctuate of the batch size.

III. Evaluation

In our evaluation, we use CIFAR10 as the training dataset and Resnet18 as the training model. Our PS structure consists of one server and two workers with different processing rates. We use a traffic dataset with 36 files from Kaggle [4]. Each file records the traffic used for time series traffic flow prediction. We use two files as the amount of training data received by two workers. To simulate varying data receiving rates, each worker reads a data amount from its file every 0.06 second and puts the training data, which is generated by randomly selecting the corresponding amount of data from CIFAR10, in the buffer. We set C to 1 and K to 120. We adjust the last full connection layer to fit the dataset. FederatedAveraging is the typical iteration round-efficient solution. We compare ABS with FederatedAveraging, in which each worker's training batch size is the size of all the data on it. In this experiment, we assume no network congestion occurs, and each node can complete data transmission within 0.1 second. Figure 4 shows the result. In this figure, ABS outperforms FederatedAveraging in terms of training efficiency and performance. In other words, data fluctuation has a relatively slight impact on the training process of ABS.

IV. Conclusion and Future Work

In this paper, we propose ABS to maintain training efficiency and performance for edge-assisted online federated learning. In future, we will consider other factors (e.g., transmission delay, training data collection) to improve our design.

References

[1] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.
[2] M. Li, *et al.*, "Scaling distributed machine learning with the parameter server," in *OSDI'14*, 2014, pp. 583–598.
[3] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," 2019.
[4] "https://www.kaggle.com/coplin/traffic."