COMP 4801

Efficiency and Robustness in Federated Learning Project Plan

Supervisor	Professor C. L. Wang
Name	Yue Yin, BEng(CS), 3035448512
	Ruijie Li, BEng(CE), 3035446992
Department	Department of Computer Science

Contents

1 Introduction **2**

2 Background **2**

2.1 Federated Learning 2

2.2 Efficiency and Robustness in FL 3

3 Objective **4**

4 Methodology **4**

5 Schedule **7**

6 Responsibility Assignment **7**

References **8**

1 Introduction

With increasing computational power, end devices like phones and IoT devices are becoming the primary computing devices for many people. High frequency of usage and powerful sensors on the devices result in an unprecedented amount of data and great opportunities for machine learning applications. However, due to the private nature and the great cost of network communication, it is infeasible to upload or store those data in a centralized location. Thus, the traditional approach of distributed machine learning, where user data is stored centrally in a data center, cannot be applied in this setting [1].

Motivated by the enormous amount of data and the need for privacy preservation, federated learning (FL) [1] is proposed to train machine learning models locally on end devices, without uploading user data. Research efforts have been made to make FL algorithms more efficient and more robust, which is also the topic of this project.

2 Background

2.1 Federated Learning

A typical federated learning system consists of servers and workers, with similar architecture to the parameter-server model [2] in the distributed machine learning problem. The federated learning paradigm involves two stages: (1) clients train models using local data and compute updates to the model, and (2) the server gathers updates from clients to obtain an aggregated update to the global model [3]. However, several key properties differentiate federated learning from traditional distributed optimization:

- **Non-IID** The training data is neither independent nor identically distributed (Non-IID) data across different end devices, as data on a particular client is collected during the usage of a particular user. Hence, any particular user's local dataset will not be representative of the global dataset [1, 4]. In traditional distributed machine learning, the dataset is usually carefully partitioned such that each partition is representative of the global dataset.
- **Unbalanced Distribution** For similar arguments, the amount of data available on each end device varies significantly [1]. In traditional distributed machine learning, the dataset is typically partitioned into blocks of similar size across the cluster.
- **Massively Distributed** The number of clients tends to be much larger than the average number of data points on each client [1]. In traditional distributed machine learning, a typical data center contains no more than a few hundred machines.

- **Resource Heterogeneity** There is significant variance in the system characteristics of end devices in computational power, training process participation ability, etc [4]. In traditional distributed machine learning, machines in a data center are usually of similar capabilities.

- **Limited Communication** End devices are normally connected through wireless networks, so communication can be unstable, slow, and expensive [1]. Though network communication could be a resource constraint in traditional distributed machine learning, machines are normally connected with much more reliable cable networks.

2.2 Efficiency and Robustness in FL

Synchronous federated learning algorithms, based on the Bulk Synchronous Parallel protocol (BSP), have been proposed [5]. In this setting, during each iteration, the server must aggregate updates from all selected clients before moving to the next iteration. In other words, clients finishing computation early must wait for other slower machines, called *stragglers* [1]. In the presence of stragglers, the overall processing time is determined by the slowest client, and a potentially large portion of time is consumed for synchronization [6], which is **inefficient**.

On the other hand, asynchronous federated learning algorithms (AFL) are designed using the Asynchronous Parallel protocol (ASP) [5], where clients proceed without waiting for each other [3]. The asynchronous nature eliminates the need for synchronization. Thus, ASP federated learning systems are usually faster than BSP systems, especially in heterogeneous environments where stragglers are common. However, when some stragglers are significantly slower than other machines, the straggler effect (also called *staleness*) arises, i.e., stragglers compute gradients based on the model parameters that are several gradient steps behind the most updated model [3, 7]. If not carefully addressed, staleness poses challenges for the **robustness** of convergence in asynchronous federated learning algorithms.

Various approaches have been explored to address the **efficiency** and **robustness** issues in FL [3–11], in both synchronous and asynchronous settings. However, we posit that the synchronous algorithms are intrinsically unscalable and inefficient, since waiting time for stragglers is inevitable, under a combinative setting of a massive number of clients, resource heterogeneity, and limited communication [3]. Thus, this project primarily focuses on asynchronous federated learning algorithms.

Asynchronous federated learning [3] algorithms have been discussed by various studies [3,8,9]. Moreover, discussions on asynchronous algorithms in traditional distributed machine learning [5, 7, 10] and synchronous FL [4, 6, 11] offer insights for AFL as well .

3 Objective

The project primarily focuses on asynchronous federated learning (AFL), which has been discussed by various studies. In this project, the objective is to thoroughly evaluate existing AFL algorithms, identify their merits and limitations, and propose new optimizations. We plan to achieve the objective through the following steps.

- A comprehensive review of related researches and detailed examination of existing proposals.
- The effectiveness and limitations of different proposals would be studied carefully, by implementation and experiments.
- Based on the experiment results, existing solutions would be compared and evaluated against each other.
- Based on insights from existing solutions and empirical researches, new solution would be proposed.
- The effectiveness of the newly proposed solution would be discussed and verified through experiments.

4 Methodology

In view of the above mentioned problems, we propose the following methodologies to optimize the efficiency of the federated learning system.

- **Stochastic Gradient Descent** SGD, as a stochastic approximation of gradient descent optimization, is often applied to high-dimensional optimization problems to speed up computations. Consider the following form of empirical risk minimization problem:

$$F(\omega) = \frac{1}{n}\sum_{i=1}^{n} f_i(\omega),$$

where $f_i(\omega)$ is the loss function of the i-th sample. Standard gradient descent will take sum of the gradients of all samples and update the weight ω with a learning rate η:

$$\omega := \omega - \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(\omega).$$

In stochastic gradient descent, instead of updating the weights based on the sum of the accumulated errors over all samples, the weights are updated incrementally for each training sample:

$$\omega := \omega - \eta \nabla f_i(\omega).$$

The stochastic approximation of the true gradient of $F(\omega)$ is the gradient at one sample. SGD can reach convergence faster because of the more frequent weight updates, and its variants like mini-batch SGD can perform better than standard stochastic gradient descent. We plan to adopt this methodology in our project as it has been proven efficient in FL algorithms like FedSGD and FedAvg. As most other models are using SGD as well, implementing our model with SGD will make comparisons more rational.

- **Asynchronous federated learning** Federated learning models come in synchronous and asynchronous forms. Both of them attempt to solve the same optimization problem:

$$\min_{\omega} F_i(\omega_i) = \frac{1}{n_i} \sum_{i=1}^{n_i} f(x_j^i; \omega_i),$$

where x_j^i is the j-th data point in the i-th worker node. At the communication round t, ω_t can be calculated by local gradient descent:

$$\omega_t := \omega_{t-1} - \eta_{t-1} \nabla F(\omega_{t-1}).$$

Synchronous FL is potentially unscalable, inefficient, and inflexible [3]. In synchronous FL, even using a selected fraction of clients, it is difficult to synchronize the selected devices at the end of each epoch due to different task scheduling policies on devices with various computational capacity and battery time. As a result, we use the asynchronous FL to address the problems caused by the synchronous flavor of federated optimization. For an asynchronous model, in the t-th global epoch, an arbitrary client will send a locally trained model ω_t to the server, where server updates the global model ω' accordingly with weight α on the global model of the previous epoch:

$$\omega'_t = \alpha \omega_t + (1 - \alpha) \omega'_{t-1},$$

The server and workers conduct updates asynchronously. The server immediately updates the global model whenever it receives a local model. The communication between the server and workers is non-blocking [3]. We use this methodology because BSP used in synchronous FL algorithms will cause "the straggler's effect" and slow down the performance by waiting for synchronization. While we are well aware of the staleness problem brought by its asynchronous nature, tackling it will be our primary focus for this project.

- **FL Framework** We will compare and try to use different existing open-source FL framework for carrying out experiments to verify assumptions. By implementing the proposed algorithm in the framework we can intuitively evaluate the efficiency, accuracy, and scalability of the system. Candidate frameworks include but are not limited to Flower [12], FATE [13], and OpenYurt.

- **Heterogeneous Clients** To evaluate an FL algorithm empirically, it is crucial to simulate a sufficient number of heterogeneous clients. We plan to use the combination of Docker [14] and Kubernetes [15]. Docker containers provide an easy interface to simulate clients with different resources, and Kubernetes helps to manage containerized applications, which, in our case, are programs running SGD to compute local updates.

- **Non-IID and unbalanced distribution** To simulate user data in typical federated learning environment, we plan to use LEAF [16], a benchmarking framework for federated learning that includes a suite of open-source federated datasets. In the experiment, elements from datasets would be unevenly partitioned and unbalancedly distributed to different clients.

- **Computational resources** It is also essential to take clients' limitations into consideration, and the computational resource is of vital importance among them. For the clients with limited computational resources, they will need a longer time to update their models locally. If the clients are using a slow network, it will also delay their updates [17]. Therefore in the experiments, we plan to simulate running FL in a batch of clients with different network conditions, computational capabilities, and data resources.

- **Datasets and machine learning models** We intend to consider FL in various application scenarios to evaluate their performance and study whether different natures of the datasets can affect the efficiency and the robustness of the FL model. Some common applications include computer vision and object detection (YOLO [18] and Faster R-CNN [19], using datasets like MNIST and CIFAR-10), language modeling with RNN, speech recognition and keyword spotting (speech command dataset).

5 Schedule

Due to the research nature of this project, it is hard to put down strict schedule for the project. Thus, we made a flexible schedule, with several major milestones.

Duration / Deadline	Scheduled work / Deliverable
Sep.1 - Oct.4 2020	Background research
Oct. 4, 2020	1st deliverable: Project plan and project website
Oct. 5, 2020 - Jan. 10, 2021	• Detailed examination of existing research • Empirically evaluate existing solutions • Compare the merits and limitations of existing solutions
Jan. 11 - 15, 2021	First presentation
Jan. 24, 2021	2nd deliverable: preliminary implementation and interim report
Jan. 25 - Apr. 17, 2021	• Propose new solutions • Prove the effectiveness of proposed solution
Apr. 18, 2021	3rd deliverable: final tested implementation and final report
Apr. 19 - 23, 2021	Final presentation
May 4, 2021	Project exhibition
Jun 2, 2021	Project Competition

6 Responsibility Assignment

All deliverable would be prepared by Ruijie and Yue together.

Ruijie and Yue did background research together.

Ruijie and Yue will examine existing researches together. For empirical study of existing solutions, Yue will focus on implementing FL algorithms, and Ruijie will focus on simulating FL environment with Docker and Kubernetes. Ruijie and Yue will prepare datasets and machine learning models together. Evaluations of and comparisons between existing solutions will be done together as well.

Ruijie and Yue will propose new solution together. When empirically evaluating and verifying the new solution, responsibility assignment will be similar as above.

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2017.

[2] Mu Li. Scaling distributed machine learning with the parameter server. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, BigDataScience '14, New York, NY, USA, 2014. Association for Computing Machinery.

[3] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization, 2019.

[4] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[5] Jiawei Jiang, Bin Cui, Ce Zhang, and Lele Yu. Heterogeneity-aware distributed parameter servers. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 463–478, New York, NY, USA, 2017. Association for Computing Machinery.

[6] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019.

[7] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-aware async-sgd for distributed deep learning, 2016.

[8] Yujing Chen, Yue Nin, Martin Slawski, and Huzefa Rangwala. Asynchronous online federated learning for edge devices, 2020.

[9] Yang Chen, Xiaoyan Sun, and Yaochu Jin. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–10, 2019.

[10] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd, 2017.

[11] Tianyi Chen, Georgios B. Giannakis, Tao Sun, and Wotao Yin. Lag: Lazily aggregated gradient for communication-efficient distributed learning, 2018.

[12] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[13] Jeff Zhang and Siddharth Garg. Fate: Fast and accurate timing error prediction framework for low power dnn accelerator design, 2018.

[14] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[15] D. Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[16] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings, 2019.

[17] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.