Front.Comput.Sci.
DOI

LETTER

Prototyping Federated Learning on Edge Computing Systems

Jianlei Yang [1,2] ✉, Yixiao Duan [1,2], Tong Qiao [1,2], Huanyu Zhou [1,2], Jingyuan Wang [1], Weisheng Zhao [2,3]

[1] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[2] Fert Beijing Research Institute, BDBC, Beihang University, Beijing 100191, China
[3] School of Microelectronics, Beihang University, Beijing 100191, China

1 Introduction

Deep learning has obtained a great success in computing technologies and has been affiliated to people's lives inseparably recent years. Nowadays, the most common approach to deploy these deep learning models is to collect the users data and train them in central servers. Nevertheless, the users data might be inevitably undermined during the process of collection and uploading, and users sensitive information might be exposed to danger. Aiming at reducing the risk of privacy leakage, Google proposed Federated Learning, which has been successfully applied to several applications, e.g. data alliance among enterprises [1], next word prediction in Gboard [2], etc.

Previous Federated Learning research has reported some applications in the protocol of enterprise cooperative pattern, which is at the level of service [1], or in the software application from user perspectives [3]. Since there will be trillions IoT devices in the future, the system efficiency and privacy protection should be taken special attention on evaluating Federated Learning algorithms in edge or mobile computing platforms. In this letter, a typical Federated Learning application is deployed onto edge computing system and shows how to efficiently implement the deep learning models on edge computing systems. Considering the communication bottleneck between different devices when exchanging and updating the learned models, Asynchronous Federated Learning (`Async-FL`) is introduced in comparison with the conventional Synchronous Federated Learning (`Sync-FL`), which is also extended from smart phone's applications [4] to IoT scenarios.

Manuscript received on June, 2019, revised on October, 2019.

E-mail: jianlei@buaa.edu.cn

2 Methodologies

Instead of uploading users data to central servers, Federated Learning (`FL`) distributes training or inference tasks to user devices for privacy protection. Usually `FL` requires several training rounds, and for each round the edge devices accept the global network, perform training tasks, submit the learned models for model aggregating. An important concern of these training rounds is the convergence rate. A typical system architecture of `FL` is proposed in [3], which is called `Sync-FL` because their partially learned models are merged synchronously. Difference with `Sync-FL`, the `Async-FL` approach is deployed in this letter. Some notations are defined as following:

- C denotes the global network on central server.
- E_n denotes the network on the n-th edge device, where $1 \leq n \leq N$.
- I is the total training rounds, and K is the epoch number for each round.
- p is the bounce rate of `Async-FL`.

2.1 Synchronous Federated Learning (`Sync-FL`)

For the i-th training round of `Sync-FL`, it has three steps [5]:

(1) Broadcasting: broadcast C to the edge devices, $E_n \leftarrow C$.
(2) Training & Submitting: train E_n on the i-th device for K epochs, then submits E_n to the server.
(3) Aggregating: after receiving the scattered models $\{E_n\}$, the server aggregates them by weighted average as a new global network C.

However, there are some obvious drawbacks in `Sync-FL`. First, the central server cannot perform model aggregation until all of the $\{E_n\}$ models have been received. Since there are usually

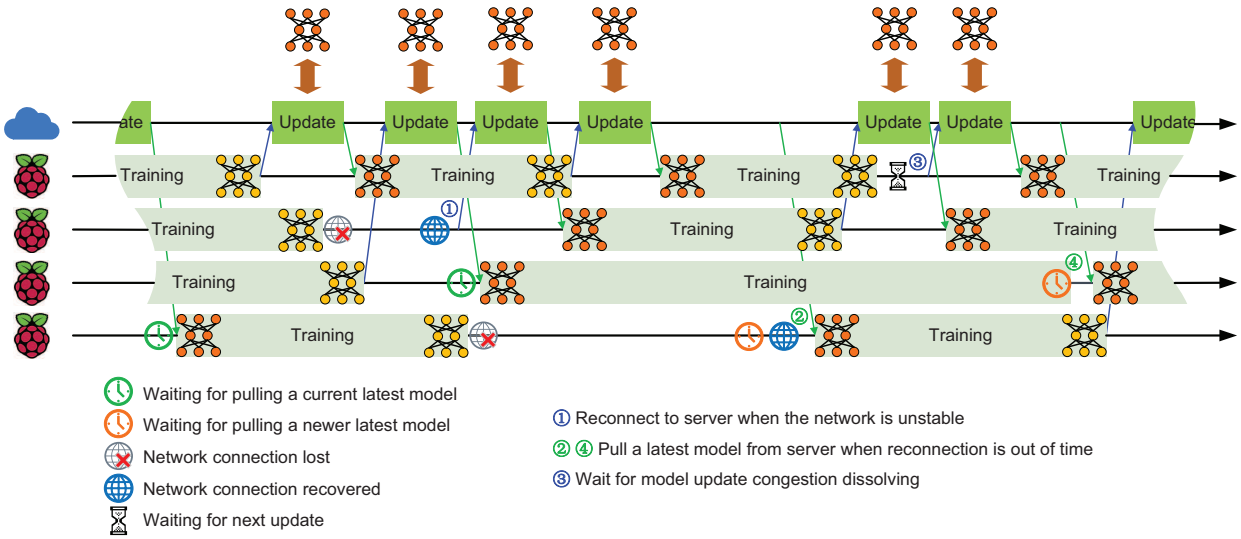


Fig. 1 System Architecture of Federated Learning on Edge Computing Platforms.

some differences in computation capacities among different edge devices, their consumed training and communication time will be different so that the total performance is determined by the worst case. Moreover, broadcasting the global network to all the edge clients at the same time will increase the communication time significantly when there are large scale of edge devices are deployed, i.e., the communication pressure is $O(N)$.

2.2 Asynchronous Federated Learning (`Async-FL`)

The drawbacks of `Sync-FL` motivate us to introduce `Async-FL`. As shown in Algorithm 1, `Async-FL` has also mainly three steps which are slightly different to `Sync-FL`. For the i-th training round:

(1) Pulling: pull the current global network C^i down from the central server to the n-th edge device as its local network $E_n^{i^*}$, then refresh the local round counter i^* to i.
(2) Training & Submitting: train $E_n^{i^*}$ on the i-th device for K epochs, then submits $\tilde{E_n^{i^*}}$ to the server.
(3) Updating: once the server receives a network $\tilde{E_n^{i^*}}$, perform bounce-update (with bounce rate p) on the global network immediately, and accumulate i.

For each training round, `Sync-FL` sends the network to all of the devices and waits for all of the trained models to aggregate. But for `Async-FL`, it only sends the latest model to the request device and then performs bounce-update for each training round. `Async-FL` could hide the latency of pulling and submitting since they are not performed simultaneously for different devices. Some certain devices could continue their training tasks when other devices are communicating with the server.

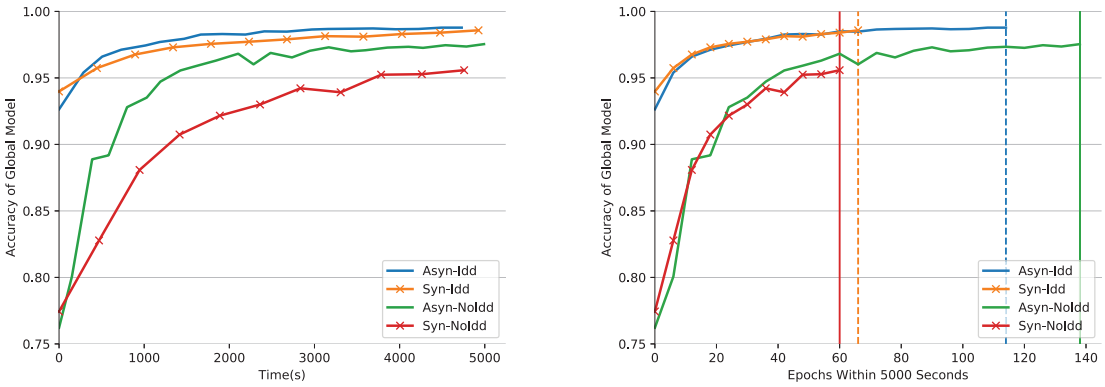
Algorithm 1 : Asynchronous Federated Learning

procedure DEVICE(n-th)
  while not *Finish* **do**
    (1) Pull global network: $E_n^{i^*} \leftarrow C^i$, $i^* \leftarrow i$
    (2) Train local network for K epochs: $E_n^{i^*} \rightsquigarrow \tilde{E_n^{i^*}}$
      Submit $\tilde{E_n^{i^*}}$ to the central server.
  end while
end procedure
procedure SERVER
  for i from 1 until I **do**
    Wait until $\exists n \in [1, N]$, $\tilde{E_n^{i^*}}$ has been submitted.
    (3) Update: $C^{i+1} \leftarrow (1-p)C^i + p\tilde{E_n^{i^*}}$
    Send current network C^{i+1} to the n-th device.
  end for
end procedure

Additionally, the server in `Async-FL` could update the global network when it receives the learned model from the faster devices with no need for waiting for other slower devices. Thus the communication pressure is reduced from $O(N)$ to $O(1)$.

2.3 System Architectures

The system design of `Async-FL` is illustrated in Fig.1. It shows how to communicate between edge devices and central/cloud server. Considering intelligent IoT devices like voice assistants, router, smart light and thermometer in a modern house, even though the computation capabilities or communication qualities of difference devices are discrepant, the model update will be not blocked in central server. Meanwhile, `Async-FL` is robust to disturbance of few devices accessibility to the server because



(a) Accuracy to training time

(b) Accuracy to total epochs

Fig. 2 Model training convergence rate of `Sync-FL` and `Async-FL` with identically or non-identically input data distribution.

the latest global model could be accessed once the lost connection is recovered. However, the server has to resolve the update request congestion when there are many devices.

3 Experiment Results and Analysis

The `Sync-FL` and `Async-FL` algorithms are implemented on a group of six Raspberry Pi 3B+ devices. The experiments are conducted by LeNet on MNIST dataset. Usually the discrepancy in computation capabilities is extremely large in real IoT conditions, while the capabilities of devices in our experiment are almost the same, so a random latency (maximum to 2 times of practical training time) is assigned into each device to emulate the discrepancy among different devices. For the input training data, they also have discrepant distributions between different devices so that two kinds of distribution are exploited: **Idd** (identically distribution, each device receives all kinds of labeled data with the same probability), and **NoIdd** (non-identically distribution, each device only receives biased labeled data).

The experimental results are shown in Fig.2 for illustrating the convergence rate with different approaches. For a certain distribution both Idd and NoIdd cases, `Async-FL` converges faster than `Sync-FL`. Since different devices usually accept different kinds of input data, NoIdd case is much more practical in real world, i.e., people take pictures of scenery which consistent with their preferences. Fig. 2(a) shows the circumstances under Idd perform better than that under NoIdd. The total exploited epochs are also compared as shown in Fig.2(b), which means the sum of the epochs from all devices. Since the lower communication frequency between server and edge devices, training tasks performed synchronously in `Sync-FL` have less total epochs than `Async-FL`, which also results slower convergence to time.

4 Conclusion

This letter prototypes a practical usage of `FL` onto edge systems. Considering the existing differences in computation abilities among different devices, `Async-FL` is proposed to relive the communication burden and tolerate the network issues. Experimental results show that `Async-FL` could behave better than traditional `Sync-FL` under certain circumstances, typically in edge system where there are significant discrepancies in computational abilities and input data distribution.

Acknowledgements Prof. Yang's work was supported in part by the National Natural Science Foundation of China (61602022), State Key Laboratory of Software Development Environment (SKLSDE-2018ZX-07), CCF-Tencent IAGR20180101 and the International Collaboration Project under Grant B16001. Prof. Wang's work was partially supported by the National Key R&D Program of China (2019YFB2101804) and National Natural Science Foundation of China (Grant No. 61572059).

References

1. Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):12:1–12:19, 2019.
2. Andrew H, Kanishka R, Rajiv M, Swaroop R, Francoise B. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.
3. Keith B, Hubert E, Wolfgang G, Dzmitry H, Alex I, Vladimir I. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.
4. Chen Y, Ning Y, Huzefa R. Asynchronous Online Federated Learning for Edge Devices. *CoRR*, abs/1911.02134, 2019.
5. Sun S, Chen W, Bian J, Liu X, Liu T. Slim-dp: A multi-agent system for communication-efficient distributed deep learning. In *Proceedings of AAMS*, 721–729, 2018.