Towards Taming the Resource and Data Heterogeneity in Federated Learning

Zheng Chai[1], Hannan Fayyaz[2], Zeshan Fayyaz[3], Ali Anwar[4],
Yi Zhou[4], Nathalie Baracaldo[4], Heiko Ludwig[4], Yue Cheng[1]

[1]*George Mason University,* [2]*York University,* [3]*Ryerson University,* [4]*IBM Research–Almaden*

1 Introduction

Traditionally, training machine learning models requires all data to be in the same place accessible to a trusted third party. However, privacy concerns and legislations such as General Data Protection Regulation (GDPR) [16] and Health Insurance Portability and Accountability Act (HIPAA) [14] inhibit transmitting data to a central place resulting in the impossibility of training machine learning models using this traditional technique. Despite these limitations, in some cases data owners would benefit from collaboratively training a model. To address this requirement, very recently *Federated Learning* (FL) has emerged as an alternative way to do collaborative model train models without sharing the training data [12] [17] [18].

In FL, each data owner, *party*, maintains its own data locally and engage in a collaborative learning procedure where only model updates are shared with an aggregator. Note that the aggregator does not have access to the data of any of the parties. Through FL, parties with relatively small datasets can learn more accurate models than they would if they had only used their own data. Examples of such scenario include a large number of individual parties providing personal data to smart phone apps and a relatively small number of competing companies within the same domain training a single model. A concrete scenario where FL has been used to collaboratively train models include Google's key board predictive model [6].

In these scenarios, parties may be very diverse. This diversity largely differentiates FL from traditional distributed learning systems such as [8,11] where a datacenter is available for careful management. Most of the times, the data parties involved in FL training have diversified training sets that may vary in size, computing power, and network bandwidth. These differences impact the FL process as we empirically demonstrate in our experimental section.

In the following, we first overview existing FL approaches. We show that stragglers are not considered by existing techniques. Then, through a preliminary study, we demonstrate the potential impact of stragglers on FL process and finally conclude with a discussion of the research problems.

2 Related Work

Existing FL approaches do not account for the resource and dataset heterogeneities [7,10,12], nor are they straggler-aware. In particular, there are two main approaches in training a FL model: *synchronous* and *asynchronous* FL.

In synchronous FL, a fixed number of data parties are queried in each learning epoch to ensure performance and data privacy. Recent synchronous FL algorithms focus on reducing the total training time without considering the straggler parties. For example, [12] proposes to reduce network communication costs by performing multiple SGD (stochastic gradient descent) updates locally and batching data parties. [7] reduces communication bandwidth consumption by structured and sketched updates. Moreover, [9] exploits randomized technique to reduce communication rounds. FedCS [13] proposes to solve data party selection issue via a deadline-based approach that filters out slowly-responding parties. However, FedCS does not consider how this approach effects the contributing factors of straggler parties in model training. Similarly, [19] proposes a FL algorithm for the use case of running FL on resource constrained devices. However, they do not aim to handle straggler parties and treat all parties as resource constrained. In contrast, we focus on scenarios where resource constrained devices are paired with high resource devices to perform FL.

Most asynchronous FL algorithms work only for convex loss and do not allow parties to drop-out. For instance, [15] provides performance guarantee only for convex loss functions with bounded delay assumption. Similarly, [3, 10] allow uniform sampling of the data parties and provide performance guarantee for convex loss functions. Furthermore, the comparison of synchronous and asynchronous methods of distributed gradient descent [4] suggest that FL should use the synchronous approach, because it is more efficient than the asynchronous approaches [12, 13].

3 Preliminary Study

We conduct an experimental study on AWS EC2 to quantify the impact of resource and dataset heterogeneity on training time of FL. We use a multi-party TensorFlow [2] setup to emulate a FL environment following the configuration settings used in [5], with δ as 0.001, ε as 8, and σ in the Gaussian mechanism as 1.0. We deploy 20 data parties to emulate a randomly picked 100-party FL environment, where each party is running inside of a Docker container. The training process

Test	# of Clients	# of CPUs	CPUs per Client
1	4	16	4
2	4	8	2
3	4	4	1
4	3	1	1/3
5	5	1	1/5

Table 1: Distribution of data parties and CPUs.

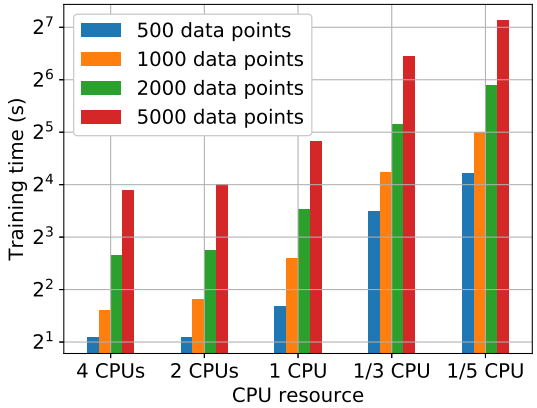

Figure 1: Per-epoch training time different CPU resources and different dataset sizes.

terminates until the accumulated privacy cost exceeds the privacy bound (δ). All the containerized parties are running on a single EC2 virtual machine (VM) instance of `m4.10xlarge` with 40 vCPUs and 160 GiB memory.

We train a CNN (Convolutional Neural Network) model on the MNIST dataset [1], which contains 60,000 28 grayscale images of ten handwritten digits. To emulate a realistic imbalanced party data distribution, we use **Non-IID** in data selection, where each party randomly selects 5 digit categories and then performs the image sampling from these 5 categories. The CNN model consists of two CNN layers and one MaxPooling layer. We use a filter size of 3 for the CNN layers and 2 for the MaxPooling layer. We also add two drop-out layers with a dropping out rate of 0.25 and 0.5, respectively. We use Adadelta for the optimizer, and accuracy as the training evaluation metric. We train the model with 8 learning epoches and measure the training time for each epoch.

Resource Heterogeneity First, we explore the impact of CPU resource heterogeneity on training time. Table 1 summarizes the parties and CPU resource distributions of 5 test groups. We reduce the total amount of CPU resources from Test 1 to 5, and within each test, each party gets an equal share of the available CPU resource. For example, in Test 1, 4 parties get allocated 16 CPU cores with 4 cores per party. Within each test group, we conduct 4 tests each with varied dataset size (sizing from 500 – 5000 data points). Figure 1 plot the average training time of one learning epoch across all data parties for each test. As shown, as the amount of CPU resources allocated to each party increases, the training time gets longer. Reducing the per-party CPU from 4 cores to 2 cores does not impact the training time much, since the CPU bottleneck is relieved with 4 CPU cores.

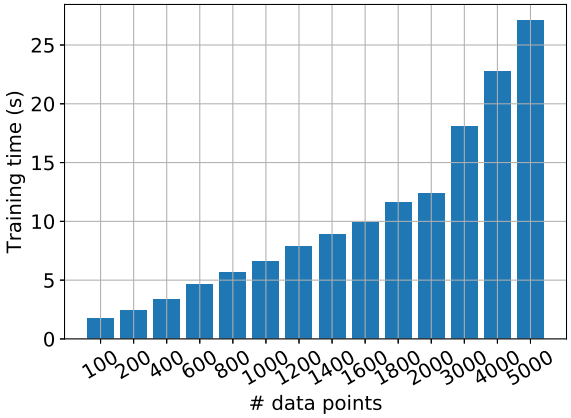

Figure 2: Per-epoch training time with different dataset sizes.

Data Heterogeneity We next quantify the impact of data heterogeneity on training time. We deploy 14 data parties, each owning a different dataset size (varying from 100–5000 data points) but with the same amount of CPU resources (i.e., 1 CPU core), to concurrently training the model. As shown in Figure 2, the training time gets linearly increased as the dataset size gets bigger. This demonstrates that data heterogeneity can significantly impact the FL system's training time.

4 Research Problems and Opportunities

Our preliminary results imply that the straggler issues can be severe under a complicated and heterogeneous FL environment. We believe that our paper will lead to discussions on the following aspects, which are the focus of our ongoing research:

P1: *How to classify parties based on their response time and then use this information for our advantage without affecting the FL process?* A naive solution can lead to misrepresentation of data, because resource constraints may be correlated with quantity/quality of data.

P2: *How to incorporate data of each party in the FL process without worrying about stragglers?* This problem is challenging because we need to make sure we do not over include or exclude certain data parties in FL process. We should be able to provide performance guarantee for general machine learning models and algorithms.

P3: *How to identify drop-out parties and mitigate the effect of drop-out data parties without affecting the ML process?* Existing approaches cannot identify drop-out parties dynamically during the FL process, and no effective method has been proposed to mitigate the information loss when drop-out happens.

Acknowledgments We thank the reviewers for their feedback. This work is sponsored in part by George Mason University, an AWS Cloud Research Grant, and a Google Cloud Platform Research Grant.

References

[1] THE MNIST DATABASE of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, 2016. USENIX Association.

[3] Inci M Baytas, Ming Yan, Anil K Jain, and Jiayu Zhou. Asynchronous multi-task learning. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 11–20. IEEE, 2016.

[4] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[5] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[6] Edwin B Kaehler. Dynamic predictive keyboard, July 7 1992. US Patent 5,128,672.

[7] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[8] Tim Kraska, Ameet Talwalkar, John C Duchi, Rean Griffith, Michael J Franklin, and Michael I Jordan. Mlbase: A distributed machine-learning system. In *Cidr*, volume 1, pages 2–1, 2013.

[9] Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, pages 1–49, 2017.

[10] Guanghui Lan and Yi Zhou. Random gradient extrapolation for distributed and stochastic optimization. *SIAM Journal on Optimization*, 28(4):2753–2782, 2018.

[11] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, Broomfield, CO, 2014. USENIX Association.

[12] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[13] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. *arXiv preprint arXiv:1804.08333*, 2018.

[14] Jacquelyn K O'herrin, Norman Fost, and Kenneth A Kudsk. Health insurance portability accountability act (hipaa) regulations: effect on medical record research. *Annals of surgery*, 239(6):772, 2004.

[15] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

[16] Colin Tankard. What the gdpr means for businesses. *Network Security*, 2016(6):5–8, 2016.

[17] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, and Rui Zhang. A hybrid approach to privacy-preserving federated learning. *arXiv preprint arXiv:1812.03224*, 2018.

[18] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, and Rui Zhang. A hybrid trust model for distributed differential privacy. *Theory and Practice of Differential Privacy*, 2018.

[19] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. *CoRR*, abs/1804.05271, 2018.