Accelerated Training via Device Similarity in Federated Learning

Yuanli Wang, Joel Wolfrath, Nikhil Sreekumar, Dhruv Kumar, Abhishek Chandra
University of Minnesota
Minneapolis, Minnnesota, USA
{wang8662,wolfr046,sreek012,dhruv,chandra}@umn.edu

ABSTRACT

Federated Learning is a privacy-preserving, machine learning technique that generates a globally shared model with in-situ model training on distributed devices. These systems are often comprised of millions of user devices and only a subset of available devices can be used for training in each epoch. Designing a device selection strategy is challenging, given that devices are highly heterogeneous in both their system resources and training data. This heterogeneity makes device selection very crucial for timely model convergence and sufficient model accuracy. Existing approaches have addressed system heterogeneity for device selection but have largely ignored the data heterogeneity. In this work, we analyze the impact of data heterogeneity on device selection, model convergence, model accuracy, and fault tolerance in a federated learning setting. Based on our analysis, we propose that clustering devices with similar data distributions followed by selecting the devices with the best processing capacity from each cluster can significantly improve the model convergence without compromising model accuracy. This clustering also guides us in designing policies for fault tolerance in the system. We propose three methods for identifying groups of devices with similar data distributions. We also identify and discuss rich trade-offs between privacy, bandwidth consumption, and computation overhead for each of these proposed methods. Our preliminary experiments show that the proposed methods can provide a 46% - 58% reduction in training time compared to existing approaches in reaching the same accuracy.

ACM Reference Format:
Yuanli Wang, Joel Wolfrath, Nikhil Sreekumar, Dhruv Kumar, Abhishek Chandra. 2021. Accelerated Training via Device Similarity in Federated Learning. In *4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3434770.3459734

1 INTRODUCTION

The increasing number of IoT and edge devices has led to exponential growth in data at the network edge. It is infeasible to send this data to remote clouds due to privacy concerns and bandwidth constraints. These concerns have led to data processing locally at the edge nodes or user devices and then sharing any features or parameters of low payload with the cloud for further processing. Distributed machine learning algorithms are a primary example of this processing strategy, which requires large amounts of data to provide accurate predictions. Federated learning (FL) [2] is the branch of distributed machine learning that depends on a federation of edge nodes for in-situ training and testing of models while preserving privacy. It has received significant attention recently and has been used in mobile applications such as search suggestion [23] and object detection [12]. Two main features differentiate federated learning from a standard distributed machine learning problem: system heterogeneity and data heterogeneity.

System Heterogeneity: Federated learning is performed on devices having heterogeneous processing capacity, network bandwidth, power, and other system resources. Many prior works have evaluated the impact of system heterogeneity on performance [5, 7, 14, 15, 20, 22]. In addition to resource differences, devices can join or leave the network during training. For example, users may switch off their mobile phones or disconnect them from the network, making them unavailable for training.

Data heterogeneity: Also known as statistical heterogeneity, data heterogeneity implies the data residing on user devices may have different probability distributions. Many factors may contribute to this variation, such as the geo-distributed nature of devices (differing time zones, events) or user behaviors. These differences violate the well-known i.i.d. assumptions in machine learning, which assume that all training samples are *independent* and come from the same joint probability distribution (*identically distributed*).

The total number of devices participating in federated learning can be in the millions [3]. Since it is not possible for all devices to be available for training simultaneously, federated learning selects a small subset of these devices in each training epoch. System heterogeneity and data heterogeneity make device selection crucial for timely model convergence and sufficient model accuracy. Therefore, methods like TiFL [5] select the set of devices participating in each training epoch based on their processing capacity, thereby accelerating training and mitigating the straggler effect. Prior works have mainly considered system heterogeneity to achieve acceptable system metrics (e.g., training speed, energy consumption) without fully considering the data heterogeneity and fault tolerance perspective. In this work, we focus on the impact of data heterogeneity on federated learning systems and explore methods for increasing performance and reliability in this setting. We conduct detailed experiments to simulate a variety scenarios for selecting devices using MNIST and MNIST-Variations datasets. Our experimental analysis demonstrates that data heterogeneity can have a significant impact on overall system performance. These results raise several research questions, including:

(1) How can we measure and quantify data heterogeneity or data similarity across multiple devices?
(2) How do we develop methods for device selection based on the quantified data heterogeneity, and how does system heterogeneity fit into the picture?
(3) How can we leverage data similarity/heterogeneity to improve the fault tolerance of the system?
(4) What are the various trade-offs for privacy, bandwidth consumption, computation overhead, training accuracy, and training speed associated with the device selection methods?

To address these questions, we propose methods for device selection that leverage the local data distributions. Our preliminary experiments show that the proposed methods can provide a 46% - 58% reduction in training time compared to existing approaches in reaching the same accuracy.

2 BACKGROUND AND RELATED WORK

Federated Learning (FL) [2] allows collaborative learning of a prediction model at user devices while preserving privacy, ownership, and locality of data. The concept enforces "bring code to data" by training models with local data on devices. During training, each participating device generates a locally optimal model update, which is propagated to the central server and systematically combined with updates from other devices to get the global update. This global update is then shared with the participating devices to further improve their local models. This sharing of model updates between central server and user devices happens over multiple iterations until model convergence is achieved. Since each device utilizes its local data for model training, FL is able to provide a personalized experience to each user. This experience, along with low latency, less power consumption, and privacy preservation has made FL very attractive in recent years. At the same time, the volatility of user devices and heterogeneity in their processing capacity, storage, network and data are some of the challenges which should be addressed in order to to build a robust FL system.

Model training in FL involves selecting a subset of devices out of the complete population of user devices. The presence of system and data heterogeneity makes device selection an important task in FL. The existing approaches for FL select devices based on random sampling [3, 6] or system heterogeneity [5]. Google recently released the design of their large scale federated learning system employed in production environment [3]. Their system first identifies the eligible devices which are idle, connected to a charging source and unmetered network (such as Wifi). Then, it randomly selects a target number of devices out of the complete set of eligible devices. TiFL [5] selects the set of devices participating in training in each iteration based on their computational speed to accelerate training and mitigate straggler effect.

Data heterogeneity across devices/locations is a significant challenge for decentralized learning systems [13, 17]. Many of the existing decentralized algorithms see a loss in accuracy when trained using heterogeneous or skewed data [6, 11, 16]. In FL, selection of devices based on random sampling or system heterogeneity can lead to high skewness as each user can potentially have a distinct data generation pattern. Hence, it is important to consider data heterogeneity while designing device selection policy.

Federated multi-task learning [9] utilizes multi-task learning [8] to model the relationships between different device data distributions so as to handle data heterogeneity. But this approach is not applicable to general non-convex deep learning models. Oort [21] selects devices based on processing power and highest utility. The cherry-picking of clients based on utility (highest loss during training) improves the convergence rate of FL models. We build upon this utility based selection method in one of the proposed methods in this work.

3 IMPACT OF DATA HETEROGENEITY

We first examine the impact of data heterogeneity on selecting devices during training in federated learning, especially at the edge. More specifically, we try to understand which subset of devices should be selected for training to ensure high accuracy while accelerating training speed and achieving fault tolerance. To that end, we conduct detailed experiments to simulate various scenarios using two datasets, and the LEAF framework [4]. Each dataset has a different type of data heterogeneity:

- **Data heterogeneity on class labels.** Here, different devices have a different distribution of class labels. For this, we use MNIST dataset [10]. All the images are 28x28 pixels hand-writing numbers labeled from [0-9].
- **Data heterogeneity within the same class.** Here, different devices have different data distributions for the same class. For example, users may write the same digit with different styles. For this, we use the MNIST Variations dataset[1], which has five different styles for each number.

For each of the two types of heterogeneity, we divide the dataset into 100 partitions and each partition is assigned to one client. We randomly select 20 clients from these 100 clients for each training epoch. We extend the distributed federated learning framework LEAF to simulate different drop patterns.

Model. We use the same Convolutional Neural Network model that is presented in LEAF. The overall accuracy is the average of the test accuracy on all devices across 1000 epochs. Wherever required, we also measure the accuracy on each device separately. Given this framework, we now turn our attention to various scenarios for device selection.

3.1 Intermittent availability of devices

In a real environment, edge devices, like mobile phones, may initially participate in training but leave after some epochs due to lack of power supply, network disconnection, or other disruption. We simulate this by dropping some devices from some training epochs while ensuring that every device has participated in at least one epoch. We use the MNIST dataset for evaluation. We adopt the setting from Zhao et al. [24] to ensure that each device contains data from at most two classes. We drop 0, 5, or 15 clients out of 20 clients in each epoch and plot the global model's overall accuracy on 100 clients after each training epoch in Fig. 1.

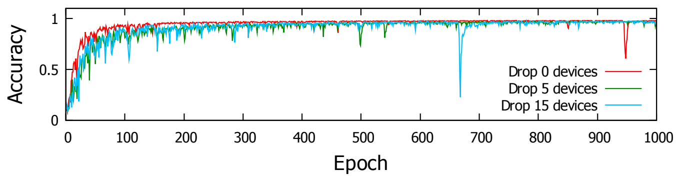

Figure 1: Accuracy of each epoch when devices drop intermittently

Observation. In Fig. 1, we see that intermittently dropping devices do not significantly impact the overall accuracy. During the initial phases of training, there is more variation in the accuracy for scenarios where the devices are dropped. However, as the training progresses, the variation continues to reduce, eventually becoming insignificant.
Conclusion. This result shows that federated learning is robust to the intermittent dropping of devices. Hence, we may not need a specially designed fault tolerance policy for such cases.

3.2 Permanent dropping of devices

In real environments, due to system heterogeneity, some devices might be significantly slower than others. Therefore, existing systems such as TiFL [5] partition and select devices based on system metrics (e.g., training speed). There is a possibility that the slower devices may never get an opportunity to participate in training. To simulate this scenario, we pre-select the set of devices which can participate in the training. Then, in each training epoch, we only select the devices from this pre-selected set. Effectively, this leads to some devices not participating in the full training.

3.2.1 Data heterogeneity on class labels. We adopt the setting from Zhao et al. [24] to partition 100 clients into 10 groups. Each group contains ten clients and will be assigned, two classes. We ensure that a device in any group will have training data only from the two classes assigned to that group, as shown in Table 1. We implement two dropping policies: 1) randomly pre-select some clients to drop 2) pre-select an entire group of devices to drop. For each policy, we drop 80 out of 100 devices and measure the trained global model's accuracy on the local test dataset of each device. The results are shown in Fig. 2.

Table 1: Partition of training data on 100 devices

Device Group No.	0	1	2	3	4
Classes	6,7	1,4	5,9	2,3	0,4
Device Group No.	5	6	7	8	9
Classes	2,5	6,8	0,9	7,8	1,3

Observation. In Fig. 2 Top, there is no drop in accuracy for any group. We conclude that if at least one client from each group participates in training, the accuracy of all devices in this group can be guaranteed. In Fig. 2 Bottom, we see that the groups which have been dropped completely experience a significant drop in accuracy. The accuracy drop for dropped groups that have some of their class labels in participating groups is less than the groups whose class labels are not present in any of the participating groups.

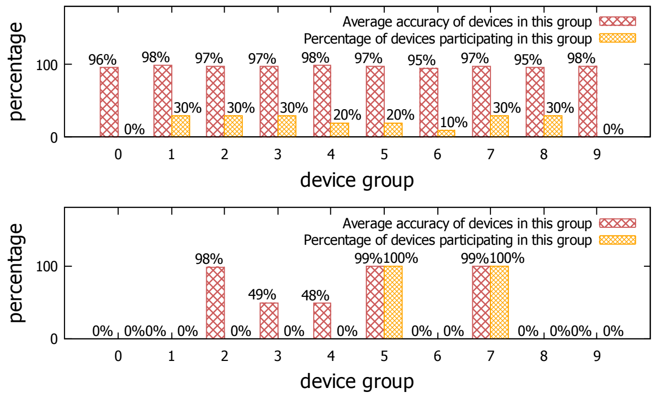

Figure 2: Top: Devices are randomly dropped permanently Bottom: Drop entire groups of devices permanently

3.2.2 Data heterogeneity within the class labels. We further consider data heterogeneity within the same class. In this experiment, we partition 100 clients into five groups. Each group contains 20 clients. All devices have all the class labels, but the devices are grouped by different image styles from MNIST Variations Dataset (See Table 2).

Table 2: Partition of training data on 100 devices

Device Group No.	0	1	2	3	4
Style	basic MNIST	Rotated MNIST + background images	MNIST + background images	Rotated MNIST	MNIST + random background

Observation. We dropped 90 clients and only left ten clients from the basic MNIST group. The evaluation result is shown in Fig. 3. The result shows that only the basic MNIST group achieved good accuracy while other groups experience varying drops in accuracy. We conclude that data heterogeneity within the same class label has a similar impact as in previous experiments.

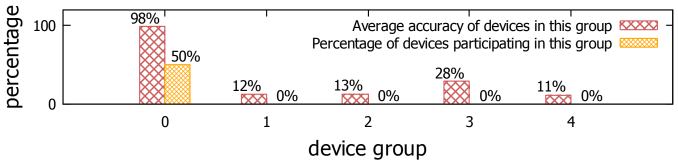

Figure 3: Average accuracy and participation rate of each group

Conclusion. Based on our analysis, we note the following:

- Federated learning is quite robust to the intermittent dropping of devices.
- The accuracy for a permanently dropped device will not drop as long as its data distribution is represented by some other devices participating in the training process.

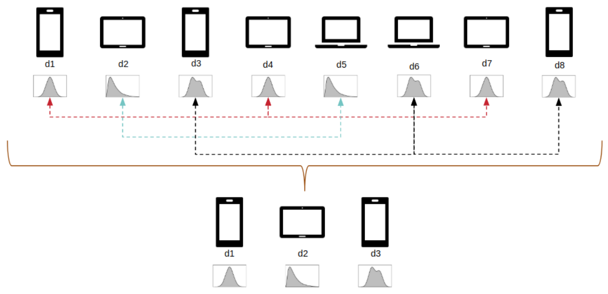

Figure 4: Data similarity across devices can be exploited for accelerated learning

We conclude that utilizing statistical metrics (along with system metrics) can significantly improve the performance and reliability in federated learning without any loss in accuracy.

4 EXPLOITING DATA HETEROGENEITY

We have shown that data heterogeneity can have a substantial impact on model accuracy. This impact is due to the violation of the i.i.d. assumption, which requires that the data residing on all edge devices is independently drawn from the same joint probability distribution. While this assumption offers nice theoretical guarantees, it is scarcely realized in practice. In this work, we seek to exploit this data heterogeneity in order to accelerate model training. To achieve this, we propose that each device asynchronously send a summary of its local data to the central node, where comparisons will be performed between these summaries to identify devices with similar data distributions. Note that the summary choice will depend on each user's performance and privacy preferences, as typical federated learning systems assume no user data leaves the edge devices.

Once the central node understands the data similarities between devices, we can leverage that information to perform training on a smaller, faster subset of the available devices. For example, if two devices, D_i, and D_j have very similar data distributions, the central node can simply schedule model training on the faster device each epoch. In this way, we can accelerate training by identifying subsets of devices with "sufficiently similar" data distributions. In the previous section, we showed that it might not be necessary to include all devices in training if the data distribution is already represented on another device. Accomplishing this objective boils down to obtaining reasonable estimates of the data distributions and computing similarity measures between them. We consider several different data summaries which could be used to accelerate training in this setting. We also discuss the bandwidth and privacy implications associated with each data summary.

Objective: Our goal is to develop methods for identifying devices with similar data distributions. Fig. 4 illustrates this process where the central node identifies devices with similar local data distributions. We simultaneously consider the privacy implications associated with each method, which is of primary importance in federated learning settings. We introduce a general model for leveraging user data to compare devices and make optimizations. To motivate our framework, consider the task of training a multiclass classifier in a federated learning setting. Define:

$$\mathcal{Z}_i = \{(X_{i,1}, y_{i,1}),\ (X_{i,2}, y_{i,2}),\ \dots,\ (X_{i,n_i}, y_{i,n_i})\}$$

to represent the training data available at each device i. Here, X_i is the matrix of input feature vectors and y_i is the associated vector of class labels. Then, each method must define and implement the following components:

1. A function $\mathcal{S}(\mathcal{Z}_i)$ which is run on each client device. This function takes the local dataset $\mathcal{Z}_i$ as input and produces a summary or distribution over the dataset which will be sent to the central node.
2. A distance function $d(\mathcal{S}(\mathcal{Z}_a),\ \mathcal{S}(\mathcal{Z}_b))$ which computes how different the summaries of $\mathcal{Z}_a$ and $\mathcal{Z}_b$ are.

The efficacy of each method depends on the selections of $\mathcal{S}$ and d, while the privacy properties of the method depend solely on the selection of $\mathcal{S}$. Our framework proceeds by having each client device compute $\mathcal{S}(\mathcal{Z}_i)$ and send it to the central server, where Algorithm 1 is executed. This algorithm groups devices by their data distribution and performs training on the fastest device in each grouping; however, this could easily be extended to selecting the top k devices from each group, or all devices that exceed a certain performance threshold.

Algorithm 1: Server task for device selection

```
Input: Device summaries $\mathcal{S}(\mathcal{Z}_i)$, distance function $d$
Result: Devices for Training this Epoch

$distMatrix \leftarrow$ Pairwise differences, $d(\mathcal{S}(\mathcal{Z}_i), \mathcal{S}(\mathcal{Z}_j))$
$groups \leftarrow$ Clustered devices based on $distMatrix$
$devices \leftarrow$ empty list
for $group$ in $groups$ do
    insert($devices$, < fastest device in $group$ >)
end
return $devices$
```

In order to motivate our selections for $\mathcal{S}$, we consider multiple ways in which the data at each device may fail to meet the i.i.d. assumptions. If the i.i.d. assumptions hold, then we can assume that all of the data generated on each device comes from some shared, joint probability distribution, $p(X, y)$. Taking this a step further, we can factor the joint distribution as follows:

$$p(X,\ y) = p(y)\ p(X \mid y)$$

which implies that the i.i.d. assumption only holds if $p(y_a) = p(y_b)$ and $p(X_a \mid y_a) = p(X_b \mid y_b)$ for all devices a, b [19]. Our method seeks to group devices by how similar their data is, i.e., which devices are likely to share a joint distribution over X and y. We consider each part of the factored joint distribution in order to estimate this. When violations of the i.i.d. assumption occur, we should be able to detect and leverage them assuming some devices have shared joint distributions, even if it is not shared across all devices.

Method 1 - Distribution of Class Labels, $p(y_i)$: As previously stated, if devices have different marginal distributions of class labels, they violate the i.i.d. assumptions. So we can attempt to use our framework to expedite the training process by letting $\mathcal{S} = p(y_i)$, the probability mass function over the class labels. We define our distance function d to be the *Hellinger distance* [18], given by:

$$H(\mathcal{S}(\mathcal{Z}_a),\ \mathcal{S}(\mathcal{Z}_b)) = \frac{1}{\sqrt{2}} \| \sqrt{\mathcal{S}(\mathcal{Z}_a)} - \sqrt{\mathcal{S}(\mathcal{Z}_b)} \|_2$$

This distance function has a few desirable properties for our application, including the tolerance for zero entries in the probability mass function and a bounded output, i.e.,

$$0 \leq H(\mathcal{S}(\mathcal{Z}_a),\ \mathcal{S}(\mathcal{Z}_b)) \leq 1$$

For this summary, we assume there are a finite number of class labels $m < \infty$. Therefore, the data size required to send this probability mass function over the network is $\Theta(m)$. Once each device sends its class label distribution to the central server, we can group devices based on how similar their label distributions are and attempt to optimize the training process based on that information.

Method 2 - Conditional Data Distribution, $p(X_i \mid y_i)$: This strategy considers using the data distribution conditioned on the class label as a summary of the local data, i.e., $\mathcal{S} = p(X_i|y_i)$. Sending this distribution exactly over the network presents problems in practice, especially if we are dealing with a continuous feature space. For example, if we have a 28x28 image, our conditional distribution is over a 784-dimensional vector space. We propose a couple of ways to handle the curse of dimensionality in this case:

- Instead of sending a huge distribution over the network, send a point estimate for the mean vector instead. This would simply be a single 784 dimensional vector for each class label, which would provide slightly stronger privacy when compared to a full distribution.
- Alternatively, we could reduce the dimension of the feature space. In this case, we might consider generating a new feature space (e.g. by using the relative frequency of individual pixel values as our features, we could reduce the dimension down to 256).

In our implementation, we performed a simple transformation of the pixel data. We reduce the dimension down to 256 by computing the relative frequency of each pixel value and then send this 256-dimensional vector across the network for each of the m labels. This summary produces a frequency distribution over the pixel values, which we then compare using the Hellinger Distance.

Method 3 - Loss based selection: This method was partially outlined in previous works [21]; here we show how it fits within our larger framework. The loss-based approach hypothesizes that devices that have similar empirical losses have similar underlying data distributions. There are certainly some situations in which this assumption does not hold, but it is a reasonable heuristic that we use for comparison. Here, we define $\mathcal{S}$ to be the empirical loss observed using the local device data on the global model. Then, the central server can simply let d be the absolute difference between the losses observed at each device.

We aggregate a handful of implications for each of the methods in Table 3. It shows the trade-offs we explored when estimating data similarity across devices with each method. Here, we assume there are n_i records at each device, which are used to perform classification across m possible labels. If dimensionality reduction is used for $p(X_i|y_i)$, we let s_i denote the dimension of the summary. The table highlights the fact that each of these methods comes with implications for the data privacy and the amount of data we are required to send.

Table 3: Trade-offs for various Dependence Estimates

Data Sent from each Device	Data Size	Privacy
Random	None	Complete Privacy
$p(y_i)$	$\Theta(m_i)$	Partial
$p(X_i \mid y_i)$	$\Theta(s_i m_i)$	Partial
Empirical Loss on Global Model	$\Theta(1)$	Stronger Privacy

5 EVALUATION

In order to evaluate the efficacy of our approach, we performed experiments with each of our proposed policies along with a random selection policy and a loss-based policy as previously discussed.

Dataset generation and device simulation: We used the MNIST handwritten digits dataset [10] as the basis for generating local training data for each device. There are $m = 10$ class labels in this dataset, representing the numbers 0-9. In order to simulate devices with similar data distributions, we allocate $x\%$ of data to a majority label, while the remaining $(100 - x)\%$ data to three random labels, per device. For our experiments, we choose $x = 91\%$ while the remaining three labels get 5%, 3% and 1% respectively.

We simulated 20 devices (2 for each class label with one being slow and other fast) and then evaluated each of the proposed methods against this partitioned dataset. The difference between fast and slow devices was simulated using a sleep function, where slow devices took 4x longer to finish each epoch.

Figure 5 shows the performance of each of the proposed methods along with a baseline strategy which randomly selects devices for participation in each epoch. Table 4 shows the exact time to convergence for each method along with the relative performance compared to the random selection policy. All of the proposed methods readily outperform the random selection policy. The fastest method uses the marginal distribution of class labels, $p(y_i)$, as the summary and reduced the training time by 58% relative to the random strategy. The conditional data distribution summary, $p(X_i|y_i)$, obtains almost the same performance as $p(y_i)$. We suspect the difference is due to the slight increase in complexity required to compute this summary. The strategy based on empirical loss was also substantially better than the random strategy with a reduced training time of 46%. While this did not perform quite as well as the other summary methods, it does have better privacy properties.

6 DISCUSSION

In section 1, we posed multiple research questions regarding data heterogeneity in federated learning. We show that by quantifying this heterogeneity, devices can be clustered based on the similarity of their data distributions. However, practitioners must carefully consider the exact choice of data summary, given the privacy and

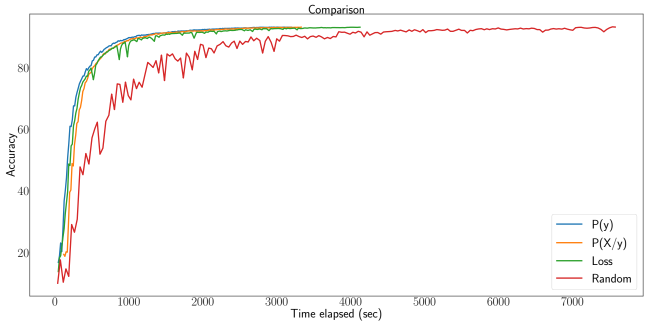

Figure 5: Full Comparison: Evaluation of all policies under consideration.

Table 4: Training time for each method and relative performance compared to random selection

Method	random	$p(y_i)$	$p(X_i\|y_i)$	loss
Duration (sec)	7578.90	3200.65	3331.22	4127.01
Reduction in Training Time	0%	58%	56%	46%

data size implications. We believe that identifying other data summaries and enumerating their properties would be a interesting research direction.

In section 3, we concluded that if there is representation of each data distribution in the training process, device dropout will not have a substantial impact on the training time and accuracy. We believe we can leverage our current clustering methods to ensure that devices from each data distribution can be represented in the training process in the event of dropout. This can be used to provide some level of fault tolerance in addition to the original goal of accelerating training.

Our work mainly focuses on clustering of devices based on the statistical information obtained about data at each device. The selection of devices from each cluster is currently based on solely the processing speed, which is only one kind of system heterogeneity. There are other factors that can be profiled like the battery power, storage capacity, stability of device and so on. We believe accounting for these factors will significantly improve the training convergence time.

7 CONCLUSION

In Federated Learning, the subset of devices taking part in the training process from a pool of millions should be selected based on certain set of features. Many of the existing works focus on random sampling [3] or processing capacity of devices [5]. Most of them are not taking into consideration the properties of data itself. In this paper we explore similarity in data distribution across different devices to select a subset for training during each epoch. We propose three methods for evaluating the similarity of device data: (1) Label distribution (2) Feature distribution conditioned over labels (3) Loss per epoch for each device. Once the devices are clustered, a device from each cluster is selected based on the processing capacity. Our experiments show 46% to 58% reduction in training time as compared to random selection to reach the same level of accuracy.

ACKNOWLEDGMENTS

This research was supported in part by the NSF under grant CNS-1717834.

REFERENCES

[1] 2014. MNIST Variations. https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits. Accessed: 2021-02-22.

[2] 2017. Federated Learning: Collaborative Machine Learning without Centralized Training Data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. Accessed: 2021-02-25.

[3] Keith Bonawitz et al. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*. mlsys.org. https://proceedings.mlsys.org/book/271.pdf

[4] Sebastian Caldas et al. 2018. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097 [cs.LG] https://arxiv.org/abs/1812.01097

[5] Zheng Chai et al. 2020. TiFL: A Tier-Based Federated Learning System. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing* (Stockholm, Sweden) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 125–136.

[6] Brendan McMahan et al. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[7] Lumin Liu et al. 2019. Client-Edge-Cloud Hierarchical Federated Learning. (2019). arXiv:arXiv:1905.06641 https://arxiv.org/abs/1905.06641

[8] Rie Kubota Ando et al. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6, Nov (2005), 1817–1853.

[9] Virginia Smith et al. 2017. Federated multi-task learning. *Advances in neural information processing systems* 30 (2017), 4424–4434.

[10] Yann LeCun et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[11] Yujun Lin et al. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* (2017).

[12] Yang Liu et al. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13172–13179.

[13] Yuanli Wang et al. 2020. Poster: Exploiting Data Heterogeneity for Performance and Reliability in Federated Learning. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. 164–166. https://doi.org/10.1109/SEC50012.2020.00023

[14] Yufeng Zhan et al. 2020. Experience-Driven Computational Resource Allocation of Federated Learning by Deep Reinforcement Learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 234–243.

[15] Zichen Xu et al. 2019. Exploring Federated Learning on Battery-Powered Devices. In *Proceedings of the ACM Turing Celebration Conference - China* (Chengdu, China) *(ACM TURC '19)*. Association for Computing Machinery, New York, NY, USA, Article 6, 6 pages.

[16] Kevin Hsieh et al. 2017. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation (NSDI'17)*. 629–647.

[17] Kevin Hsieh et al. 2020. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*. PMLR, 4387–4398.

[18] T. Kailath. 1967. The Divergence and Bhattacharyya Distance Measures in Signal Selection. *IEEE Transactions on Communication Technology* 15, 1 (1967), 52–60.

[19] Peter Kairouz et al. 2019. Advances and Open Problems in Federated Learning. *CoRR* abs/1912.04977 (2019). http://arxiv.org/abs/1912.04977

[20] Dhruv Kumar et al. 2019. DeCaf: Iterative Collaborative Processing over the Edge. In *HotEdge*.

[21] Fan Lai et al. 2020. Oort: Informed Participant Selection for Scalable Federated Learning. *arXiv preprint arXiv:2010.06081* (2020).

[22] Shiqiang Wang et al. 2018. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 63–71.

[23] Timothy Yang et al. 2018. Applied Federated Learning: Improving Google Keyboard Query Suggestions. arXiv:1812.02903 [cs.LG] https://arxiv.org/abs/1812.02903

[24] Yue Zhao et al. 2018. Federated Learning with Non-IID Data. arXiv:1806.00582 [cs.LG] https://arxiv.org/abs/1806.00582