LEARNINGS FROM FEDERATED LEARNING IN THE REAL WORLD

Christophe Dupuy [1], *Tanya G. Roosta* [1], *Leo Long* [1], *Clement Chung* [1], *Rahul Gupta* [1], *Salman Avestimehr* [1,2]

[1]Amazon Alexa AI, Cambridge, MA, USA; [2]University of Southern California (USC), CA, USA

ABSTRACT

Federated Learning (FL) applied to real world data may suffer from several idiosyncrasies. One such idiosyncrasy is the data distribution across devices. Data across devices could be distributed such that there are some "heavy devices" with large amounts of data while there are many "light users" with only a handful of data points. There also exists heterogeneity of data across devices. In this study, we evaluate the impact of such idiosyncrasies on Natural Language Understanding (NLU) models trained using FL. We conduct experiments on data obtained from a large scale NLU system serving thousands of devices and show that simple non-uniform device selection based on the number of interactions at each round of FL training boosts the performance of the model. This benefit is further amplified in continual FL on consecutive time periods, where non-uniform sampling manages to swiftly catch up with FL methods using all data at once.

***Index Terms*—** Federated Learning, NLU

1. INTRODUCTION

Deep neural networks have shown great promise in a variety of application domains. The data required for training these models is often generated via devices interaction with edge devices. In a traditional training framework, the data is sent to a central server, where the model gets trained and periodically deployed on the edge devices. The desire to strengthen data privacy and security has driven a surge of interest in developing training methodologies that can operate on local data that is not transmitted to the cloud. A promising framework is Federated Learning (FL), where each device performs local training using its own data and only sends updated model parameters to the server. The server aggregates these updates to build the new model and sends the latest model back to each device. In order to provide an FL solution that scales to millions of users, in most algorithms (e.g., `FedAVG` [1]), the central server first selects a subset of devices uniformly at random, then computes a weighted average of their parameter updates to construct the global model. In real world scenarios with heterogeneous users, the contribution of different devices might significantly vary and uniform sampling might not provide the best strategy. As such, device selection for model aggregation is a critical component of FL algorithms.

Most recent works that study device selection in FL (e.g., [2, 3]) have focused on gradient-based methods, in which the local updates of the devices are compared to decide the selection and aggregation. These methods require the computations to be carried out across all devices in each round. In this paper, we investigate simpler device selection processes that are based on "device activity" (i.e. selecting devices based on their number of available training samples). Given the FL server already has access to this information, no additional computation/communication overhead is incurred. We also study temporal effects of training data in FL, and look at how different device selection and training strategies affect the performance of the model. Most FL papers present results on "static" datasets where the time dimension is ignored.

Our contributions: (1) we present the first study of simple device selection based on device activity applied to real-world data, exploring several sampling strategies and show that non-uniform sampling outperforms standard uniform sampling; (2) we study continual FL to investigate the realistic temporal aspects of FL and address limited data storage on-device. We demonstrate that non-uniform strategies combined with continual learning swiftly catches up with FL methods using all data at once (at the expense of longer delays between updates).

2. RELATED WORK

Standard FL algorithms such as `FedAVG` [1] and `FedOpt` [4]) have been widely studied. `FedAVG` has been proved to converge under a large set of assumptions (e.g., non- I.I.D. data distributions) [5, 6, 7, 8]. The performance of these algorithms has been evaluated in CV (e.g., [1]), NLP (e.g., [9]), and graph-structured data (e.g., [10]). Recently, several works have looked at leveraging the results of previous local device update to compute the sampling probability for the next round. [11] propose a probability based on the "importance" of each device, which is mostly determined by the norm of the latest local update. In [12], the authors show that sampling devices with higher losses leads to faster convergence compared to vanilla `FedAVG`. Similarly, other methods leverage the latest local gradients to update the device sampling probability [2, 3]. These methods require all the devices to perform local updates which increases the communication overhead. Continual learning (e.g., [13, 14, 15]) has become increasingly important for learning across a sequence of rounds and tasks. More recently, continual learning has also been modeled in FL to study how one can leverage learned knowledge from different devices with different tasks [16]. In this work, we focus on temporal aspects of continual FL, meaning leveraging the knowledge from past interactions in the future training.

3. PROBLEM STATEMENT

Framework. In standard machine learning, the central server stores the n user generated entries $X = (x_i)_{i=1\dots n}$ and the objective is to minimize the average loss $\mathcal{L}$ over these n inputs: $\min_w \mathcal{L}(X, w) = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, w)$. In practice, the number of inputs n is large and $\mathcal{L}$ is minimized with stochastic gradient descent (SGD) [17] where the gradient with respect to a random batch of inputs B_t is used to update the parameters w: $w_{t+1} = w_t - \eta\nabla_w \left(\frac{1}{|B_t|}\sum_{i\in B_t} \ell(x_i, w)\right)$. In FL, the data is not available to the central server to generate random batches of data B_t. If each of the M devices contains $X^m \subset X$ inputs with $|X^m| = n_m$, $m = 1, \dots, M$, the central objective can be rewritten: $\mathcal{L}(X, w) = \frac{1}{n}\sum_{m=1}^{M} n_m \mathcal{L}^m(X^m, w)$, with $\mathcal{L}^m(X^m, w) = \frac{1}{n_m}\sum_{x\in X^m} \ell(x, w)$. In `FedAVG`, this objective is minimized by iteratively sampling a subset of devices $S_t \subset \{1, \dots, M\}$, requiring the selected devices to optimize their own objective $\mathcal{L}^m(X^m, w_t)$ locally using the latest version of the model w_t as the starting point. The updated local parameters w_{t+1}^m are sent to the central server for aggregation via: $w_{t+1} = \sum_{m\in S_t} \frac{n_m}{N_t} w_{t+1}^m$, where $N_t = \sum_{m\in S_t} n_m$ is the total number of inputs on the selected devices S_t.

Device Selection. In standard FL training, the device selection is made uniformly at random. In a real world scenario with non-I.I.D. data, different device selection strategies could help boost the performance of the resulting model. Previous works used gradient-based selection methods [2, 3] where gradients are computed for all devices before selecting a subset. We look at a simpler/less computationally expensive approach using the number of inputs per device (n_m). We want to understand: 1) change in performance if we put more weight on the active devices, 2) whether we can limit ourselves to only a subset of devices. In our experiments, we explore several selection strategies for S_t that are a function of the number of inputs on each device, namely $\mathbb{P}[m \in S_t] \propto f(n_m)$, with $f(x) \in \{1, \log(x+1), \sqrt{x}, x, \frac{1}{\log(x+1)}\}$, where $f(x) = 1$ corresponds to uniform sampling. We experiment with two additional functions referred to as `heavy` (resp. `light`) to restrict the optimization to the most (resp. least) active devices: $f(n_m) = 1$ if m is in top 20% most (resp. least) used devices, $f(x) = 0$ otherwise.

Continual Learning. In the *one-shot FL* setting, the data is static and used all at once during training. This approach might not be applicable to real-world scenario where users generate data over time due to limited storage on device. In *continual FL*, we consider a dataset generated over a time period T, which is divided into consecutive segments of size Δt. For each segment, we train a model on the data for that segment, starting from the end model of the previous segment. For the first segment, we use the same starting point as for *one-shot FL*. Similar to *one-shot*, we compare the different device selection strategies in a *continual* setting.

Our experiments address the following: **Device selection:** 1) Is there a non-uniform sampling strategy that outperforms uniform sampling? 2) Can we obtain a similar performance as the standard uniform sampling by restricting training to the most active devices? **Temporality:** 1) For *one-shot FL*, does an increase in the amount of historical data have a positive effect on the performance on the next interactions? 2) Does *continual FL* compete with *one-shot FL*, and is there a device selection strategy that boosts *continual FL* performance?

4. EXPERIMENTAL SETTING

Model: BERT-*base* model [18] for the masked language modeling (MLM) task that is pre-trained model on the Common Crawl dataset [19].

Data: 12 months of automated transcription of Alexa user request data, that is de-identified to remove information that connects the data to the users. We randomly select 10,000 devices and their utterances. In all experiments, the test set corresponds to the last month of data. We repeat the process 3 times to generate different datasets and evaluate our approaches. In all the figures, the standard deviation across the 3 datasets is displayed.

One-shot FL: The FL models are trained on $|T| = 3$ months (months 9 to 11 of the 12 months data) $|T| = 6$ months (months 6 to 11) and $|T| = 11$ months (months 1 to 11).

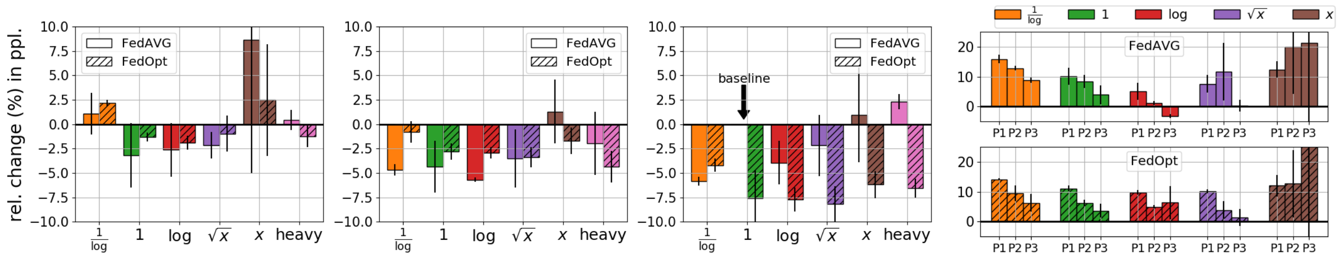
Continual FL: 6 months of data (months 6 to 11 of the whole data) is split into 3 parts of $\Delta t = 2$ months. For each time period, the last state of the model from the previous period is used for initialization.

FL methods: We compare two of the most studied FL algorithms: `FedAVG` [1] and `FedOpt` [4]. For both, we select 800 devices per round and each device performs one local epoch over their data, with standard central aggregation.

Evaluation: We use the perplexity metric and study the impact across the different devices by splitting the test set into deciles of devices, based on their number of utterances. For each decile, the perplexity is computed on all the utterances by devices in that decile. We can't present absolute performance values, instead we present relative change in % in perplexity (referred to as RCP from this point on) compared to the perplexity on the entire test set for the baseline model.

5. RESULTS

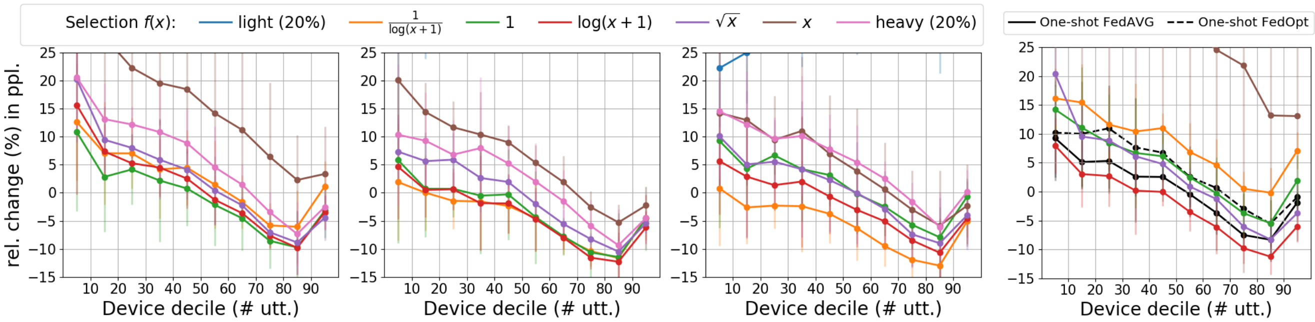
One-shot FL – Figures 1a, 2a and 3a. The baseline for the RCP computation is the standard *one-shot* `FedAVG`



(a) *One-shot FL*. **Left**: 3 months; **middle**: 6 months; **right**: 11 months.

(b) *Continual FL*. Performance after each of the 3 time periods P1, P2, P3.

Fig. 1: RCP compared to the the baseline model on the whole test set. (1a) Baseline: standard `FedAVG` ($f(x) = 1$) on 11 months of data; (1b) Baseline: *one-shot FL* `FedAVG` and $f(x) = \log(x + 1)$ on 6 months.



(a) *One-shot FL*. **Left**: 3 months; **middle**: 6 months; **right**: 11 months.

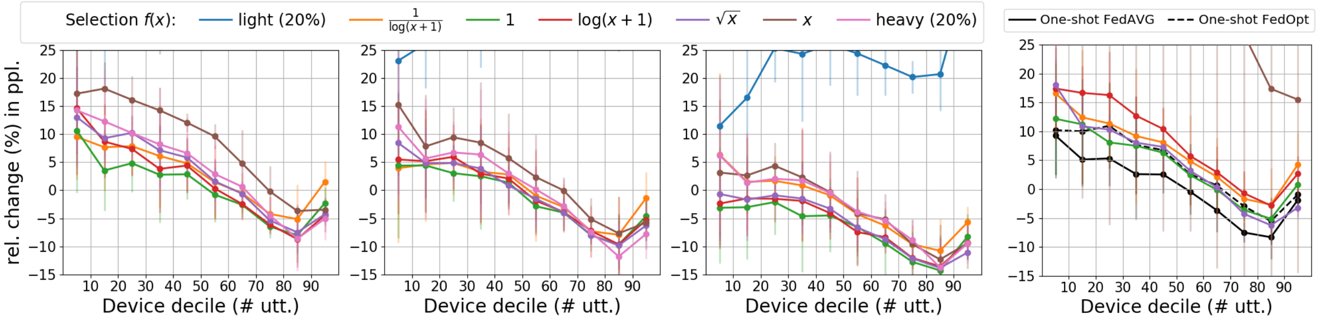
(b) *Continual FL*.

Fig. 2: `FedAVG`. RCP compared to the the baseline model on the whole test set. (2a) Baseline: standard `FedAVG` on 11 months of data; (2b) Baseline: *one-shot FL* `FedAVG` and $f(x) = \log(x + 1)$ on 6 months of data; `o-s` stands for "*one-shot*".

trained on 11 months of data. Increasing the amount of training data is clearly beneficial for `FedOpt`, but not as much for `FedAVG` (Figure 1a). As a result, `FedOpt` outperforms `FedAVG` for most of the selection strategies with 11 months of training data. It's the other way around for $|T| \in \{3\,\text{months}, 6\,\text{months}\}$. Interestingly, standard `FedAVG` using the entire 11 months of training data performs worse than almost any other strategy, possibly due to a combination of overfitting and sub-optimal selection strategy. In Figures 2a and 3a, for every method, the perplexity is lower for devices with a higher number of utterances. Every curve sees an uptick in perplexity for the last decile. From the heaviest devices (especially the top 1%), some also have a high number of utterances in the test set and are harder to predict.

`FedAVG`: the different sampling strategies significantly affect the performance of the resulting model. Overall, for shorter time range (left 2 plots in Figure 2a), using the standard uniform sampling ($f(x) = 1$) provides one of the best strategies, on par with the log-based sampling. However, for longer time horizon (right plot in Figure 2a), non-uniform sampling provides much better results. The model where the probability to select device m is proportional to $f(n_m) = \frac{1}{\log(n_m+1)}$ outperforms all the other models trained using `FedAVG`. In this strategy we tend to select lighter devices with higher probability. Conversely, selection strategies that are restricted to a subset of devices (heavy or light devices only and $f(x) = x$) perform poorly. The selection $f(x) = x$ performs worse than when restricting sampling to the top 20% devices because of the distribution of the devices: their number of utterances varies wildly across the dataset (with a factor of $> 10^4$), and choosing $f(x) = x$ implies the model is likely to select the same few devices (less than 20% total) having much more utterances than the rest. In terms of volume, it is not beneficial to extend the size of the data for `FedAVG`, except for the best strategy $f(x) = \frac{1}{\log(x+1)}$ that performs much better with 6 months over 3 months of data, but only slightly better when using 11 months compared to 6 months. In Figure 2a, we see that more training data mostly benefits light devices. For instance, with 3 months of data and $f(x) = \frac{1}{\log(x+1)}$, the average RCP for the first decile is around 13% and drops to around 1% when using 11 months of data. For the last decile, RCP "only" drops from 2% to -5%.

`FedOpt`: contrary to `FedAVG`, the performance is similar across the selection strategies, except for selection restricted to `light` devices that performs poorly in general. Restricting the training to the top 20% devices fares as well as the other



(a) *One-shot FL*. **Left**: 3 months; **middle**: 6 months; **right**: 11 months.

(b) *Continual FL*.

Fig. 3: `FedOpt`. RCP compared to the the baseline model on the whole test set. (3a) Baseline: standard `FedAVG` on 11 months of data; (3b) Baseline: *one-shot FL* `FedAVG` and $f(x) = \log(x + 1)$ on 6 months of data; `o-s` stands for "*one-shot*".

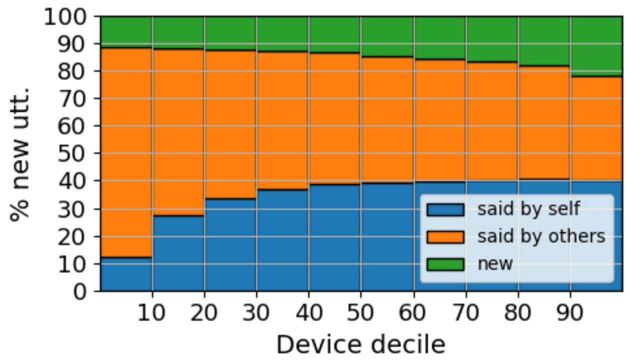


Fig. 4: Average repartition of new utterances over a period.

strategies involving all the devices. As expected, this strategy outperforms on heavier devices and underperforms on lighter devices which have not been seen by the model. From Figure 3a, uniform sampling provides better performance on light devices, but other strategies provide better performance on the overall test set ($f(x) \in \{\log(x + 1), \sqrt{x}\}$).

Continual FL – Figures 1b, 2b and 3b. The baseline is the best *one-shot* model on the same 6 months: `FedAVG` trained with $f(x) = \log(x + 1)$. The advantage of *continual FL* is the possibility to store less data on device but still train a model on the entire time period we consider. From Figure 1b, models trained with most selection strategies improve performance along the time periods, except $f(x) = x$. Several selection strategies with *continual FL* display less than 5% relative degradation compared to the best *one-shot FL* model after the third period of training (P3). *Continual FL* consistently outperforms the *one-shot FL* baseline for `FedAVG` with $f(x) = \log(x+1)$. However, no other selection strategy beats the baseline with *continual FL*, even if $f(x) = \sqrt{x}$ is on par with the baseline for both `FedAVG` and `FedOpt`. Similar to *one-shot FL*, the difference in performance across the selection strategies is larger for `FedAVG` than for `FedOpt`.

`FedAVG`: The gap between *one-shot FL* and *continual FL* across device deciles for $f(x) = \log(x + 1)$ (black and red curves in Figure 2b) indicates that *continual FL* benefits the heavy devices. This is expected as heavier devices are more likely to have utterances in all the 3 periods used for training.

`FedOpt`: Even if all the selection strategies used with `FedOpt` perform worse than the *one-shot* `FedAVG` baseline, they are consistently on par with the *one-shot* `FedOpt` model across device deciles. Similar to `FedAVG`, *continual FL* benefits heavy devices more. For instance, *continual FL* with `FedOpt` and $f(x) = \sqrt{x}$ performs worse than the *one-shot* `FedOpt` baseline on the first device deciles, but outperforms this method on the last deciles.

Insight. We look at the repartition of utterances for a given period compared to all the previous ones across device deciles. For a given device's set of unique utterances over a time period, we compute 1) the proportion of utterances already seen by this device; 2) the proportion of new utterances for this device but seen by any other device in a previous period of time; 3) the proportion of completely new utterances. In Figure 4, these metrics are averaged over periods P2 and P3. We observe that heavy devices explore more than light devices, who are mostly "catching up". For learning, it seems more beneficial to sample heavier devices who bring additional knowledge.

6. CONCLUSION

We studied popular FL algorithms on non-I.I.D. data borrowed from a real word setup. Through our experiments, we show that non-uniform device selection outperforms standard uniform sampling of FL and boosts performance for most devices. We also compare the selection strategies on different amounts of data and observe that `FedOpt` benefits from more data, but not `FedAVG`. Finally, we compare *one-shot FL* with *continual FL* where we run multiple rounds of training on smaller time periods, allowing for lower storage capacity. We show that *continual FL* outperforms the *one-shot* strategy in some setting, and is overall most beneficial for heavy devices.

7. REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes, "Diverse client selection for federated learning: Submodularity and convergence analysis," in *ICML 2021 International Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2021.

[3] Lokesh Nagalapatti and Ramasuri Narayanam, "Game of gradients: Mitigating irrelevant clients in federated learning," May 2021, vol. 35, pp. 9046–9054.

[4] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan, "Adaptive federated optimization," in *ICLR*, 2021.

[5] Haibo Yang, Minghong Fang, and Jia Liu, "Achieving linear speedup with partial worker participation in non-iid federated learning," 2021.

[6] Yichen Ruan, Xiaoxi Zhang, Shu-Che Liang, and Carlee Joe-Wong, "Towards flexible device participation in federated learning," 2021.

[7] Elsa Rizk, Stefan Vlaski, and Ali H. Sayed, "Dynamic federated learning," 2020.

[8] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama, "Federated learning based on dynamic regularization," in *ICLR*, 2021.

[9] Bill Yuchen Lin, Chaoyang He, Zihang Zeng, Hulin Wang, Yufen Huang, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr, "Fednlp: A research platform for federated learning in natural language processing," 2021.

[10] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Yu Rong, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr, "Fedgraphnn: A federated learning system and benchmark for graph neural networks," .

[11] Wenlin Chen, Samuel Horvath, and Peter Richtarik, "Optimal client sampling for federated learning," 2020.

[12] Yae Jee Cho, Jianyu Wang, and Gauri Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," 2020.

[13] Sebastian Thrun, "Lifelong learning: A case study," Tech. Rep. CMU-CS-95-208, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1995.

[14] Jonathan Schwarz, Jelena Luketina, Wojciech M. Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell, "Progress & compress: A scalable framework for continual learning," in *ICML*, 2018.

[15] Steven C. Y. Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen, "Compacting, picking and growing for unforgetting continual learning," 2019.

[16] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang, "Federated continual learning with weighted inter-client transfer," in *ICML*, 2021.

[17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. 2010, pp. 177–186, Springer.

[18] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[19] Common Crawl Foundation, "Common crawl data," `https://commoncrawl.org/`, 2010.