Jointly Learning from Decentralized (Federated) and Centralized Data to Mitigate Distribution Shift

Anonymous Author(s)
Affiliation
Address
email

Abstract

With privacy as a motivation, Federated Learning (FL) is an increasingly used paradigm where learning takes place collectively on edge devices, each with a cache of user-generated training examples that remain resident on the local device. These on-device training examples are gathered in situ during the course of users' interactions with their devices, and thus are highly reflective of at least part of the inference data distribution. Yet a distribution shift may still exist; the on-device training examples may lack for some data inputs expected to be encountered at inference time. This paper proposes a way to mitigate this shift: selective usage of datacenter data, mixed in with FL. By mixing decentralized (federated) and centralized (datacenter) data, we can form an effective training data distribution that better matches the inference data distribution, resulting in more useful models.

1 Introduction and Background

Federated learning (FL) [McMahan et al., 2017] is a machine learning setting where multiple 'clients' (typically, edge computing devices like mobile phones) collaborate in solving a machine learning problem, under the coordination of a central server. Each client caches raw training data locally, and the data are never exchanged or transferred. Instead, focused updates intended for immediate aggregation are used to achieve the learning objective [Kairouz et al., 2019]. Providing strong privacy protection is a major motivation for FL. Storing data locally rather than replicating it in the cloud decreases the attack surface of the system, and the use of focused ephemeral updates and early aggregation follow the principle of data minimization [White House Report, 2013]. Stronger privacy properties are possible when FL is combined with technologies such as differential privacy (DP) and secure multiparty computation (SMPC) [Wang et al., 2021].

FL also typically delivers utility improvements such as increased task accuracy, because the training examples gathered in situ by edge devices are reflective of actual inference serving requests. For example, a mobile keyboard next word prediction model can be trained from actual SMS messages, yielding higher accuracy than a similar model trained instead on a proxy document corpus. Because of these benefits, FL has been adopted and used to train production models for a variety of mobile phone applications, such as keyboard suggestions and audio keyword spotting [Hard et al., 2018, Ramaswamy et al., 2019, Apple, 2019, Hard et al., 2020].

While FL reduces train vs. inference distribution skew, it doesn't remove it completely. The FL training examples may still not fully represent the inference distribution, due to situations such as:

- Label-biased example retention skewing the cached dataset away from the inference distribution. The nature of users' interactions with their devices often deters the caching of negative examples. For example, a mobile phone's photo cache reflects photographic subjects a user is interested in. It generally does not contain photos of subjects a user is *not* interested in

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

(such photos were either not taken, or were taken and deleted). If desiring e.g. a model that predicts interest in a camera scene, the lack of negative examples (i.e., the label imbalance) would be a significant impediment to training a binary 'interest' classifier.

- A model that trains on one class of edge devices but is used for inference on other classes of devices as well. E.g., an audio model may train on mobile phones but also be deployed for inference on smart speakers (with different microphone characteristics), or train on high-end phones but also be deployed for inference on low-end phones (with different user characteristics).
- Safety critical applications of AI. Risky scenarios requiring extremely accurate inference behavior are (hopefully) infrequently encountered by users in the course of typical day-to-day device interaction (and thus lacking in the federated training examples). E.g., the intelligent driving assistance prompts to the driver of an automobile skidding off a road in heavy rain.

How can the benefits of FL (improved privacy and utility) be achieved while overcoming the remaining train vs. inference distribution skew? This paper proposes a solution: mixing data from an additional datacenter dataset into a FL training process, to afford a 'composite' set of training data that better matches the inference distribution. The process for mixing is not trivial, given the stringent privacy requirements of FL. In traditional datacenter-based learning, a ML engineer can simply concatenate two (or more) sets of training data into one, randomly shuffle, and then draw batches of mixed examples. But this level of fine-grained interaction with training examples is precluded in FL, where the on-device examples (containing user private information that reveals identity) stay on device and can never be handled individually by the ML engineer [Augenstein et al., 2019].

This paper discusses strategies for training models which meet the private data access constraints imposed by FL while also leveraging additional data available at the server, to achieve models that perform better on the inference distribution. Section 2 presents several such mixing strategies, Section 3 compares their performance at several modeling objectives, and Section 4 discusses some perspectives and logical extensions in mixing decentralized and centralized data.

2 Strategies to Learn on Decentralized and Centralized Data

We now consider three strategies that perform FL while mixing in additional data from a centralized dataset. We use FEDAVG [McMahan et al., 2017] (Algorithm 1), a ubiquitous FL algorithm, as the learning approach of reference. Each of the strategies below is a modification of FEDAVG. The algorithms for the strategies are presented in Appendix A. We demonstrate the application of the strategies to an example problem using the CelebA dataset in Appendix B.

Synchronous Parallel Training In synchronous parallel training, we start with a global model, and then separately in parallel perform a round of FEDAVG (with the decentralized data) and steps of training (with the centralized data). This yields two updated versions of the model, one via FL and one via central training. After every round, we take these two versions of the model, merge them together (e.g., average the weights) to form a new global model, and repeat. See Algorithm 2 for details.

Example Transfer In example transfer, we perform standard FEDAVG with one additional aspect. For each client participating in the FL round, we sample some training examples from the centralized data and send these to the clients (along with the latest version of the global model). These datacenter examples are merged and shuffled with the local client examples, and then batches of 'mixed' data are drawn when performing the client training portion of standard FEDAVG. See Algorithm 3 for details. In this way, the client optimization accounts for both decentralized data and centralized data as it proceeds towards a minima.

Gradient Transfer In gradient transfer, at the start of a round, we first take the latest version of the global model and calculate a gradient w.r.t. a batch of datacenter training examples. This gradient is then sent to the participating clients (along with the latest version of the global model). During each step of client training, we calculate a gradient w.r.t. batches of client data examples, sum this gradient with the 'augmenting' datacenter gradient, and then use the summed gradient when performing client

optimization. See Algorithm 4 for details. In this way, the client optimization is 'biased' in a direction such that it is minimizing of both decentralized data and centralized data.

Related Work In terms of algorithmic approach, there are parallels between some of the above strategies and algorithms for addressing inter-client data heterogeneity in FL. For example, the example transfer strategy is reminiscent of a data-sharing strategy proposed in Zhao et al. [2018]. That work aimed to address the problem of optimization on non-IID client datasets, whereas we apply the concept to mixing between the federated clients' overall (decentralized) distribution and a separate centralized distribution. Similarly, the gradient transfer strategy is inspired by the SCAFFOLD [Karimireddy et al., 2020b] and Mime [Karimireddy et al., 2020a] algorithms. These algorithms calculate a gradient term that is reflective of the client population as a whole, and transmit it to the clients each round to reduce the variance in the respective client updates. By contrast, the gradient term here is derived from the centralized data, and not the decentralized federated data.

3 Comparisons

There are tradeoffs in selecting between the strategies above, based on comparative accuracy, data transfer payload size, and privacy considerations.

Accuracy Differences exist in the resulting accuracy of a classifier trained by the various mixing strategies. This is shown for example in Figure 1, for the smile classifier described in Appendix B. Example transfer could be expected to perform best, effectively equivalent to a scenario where there were no label imbalance issues and one could just perform pure FL with smiling *and* unsmiling faces. Gradient transfer could be expected to sometimes perform worse than example transfer, for the reason that the gradient w.r.t. centralized data is calculated once and then frozen over the course of the clients' optimization steps. It may grow 'stale' after several steps, as compared to example transfer where the gradient w.r.t. centralized data is calculated fresh with every step of optimization (because centralized data examples exist on device). In general, further analysis is necessary to understand the relative convergence behavior of these mixing strategies. Also, note that parallel training has a larger hyperparameter space to explore in understanding its optimal converge settings.

Transfer payload size Example transfer and gradient transfer both require transmitting additional information (along with the latest model checkpoint) from server to clients at the start of each federated round. Naively, example transfer adds an amount to the payload of the order of example size times number of examples, and gradient transfer doubles the transfer payload (as the gradient is the same size as the model). Studying compression strategies would be a useful future research exploration, in particular for gradient transfer, as it seems likely that not all gradient terms are equally necessary. Parallel training incurs no additional transfer cost above pure FL.

Privacy None of the three mixing strategies send additional information back from the clients to the server, so there are no added concerns (above those in standard FEDAVG) with respect to protecting user private data on the clients. The same tools for protecting user data during FL, like user-level DP [McMahan et al., 2018], can be brought to bear. Both example transfer and gradient transfer broadcast information about the centralized training data from the server to the clients, but in many scenarios the centralized data is not privacy sensitive (e.g., it is a public corpus). Were the centralized training data to also be privacy sensitive, then that imposes some constraints on what is possible. Example transfer would be precluded, unless heavily modified to an intricate approach that utilizes DP generative models to synthesize privacy-preserving examples [Augenstein et al., 2019]. Gradient transfer is less directly revealing of the raw centralized examples then example transfer, but recent work has shown that reconstructing sensitive information from gradients alone is indeed possible [Dang et al., 2021]. However, the same work demonstrates methods for defending against such an attack, which could be adopted if applying gradient transfer to a mixing problem involving sensitive centralized data.

4 Discussion and Further Directions

This paper has described and compared basic mixing strategies that result in models that perform better at inference time. We now consider some perspectives on mixing centralized and decentralized data, which inform extensions to the basic strategies presented here.

One perspective on mixing: it is a merging of 'wisdom of the crowd' (from decentralized, user private data) with 'wisdom of laboratory experts' (from an organized, purposeful data collection exercise). The 'crowd' training distribution conveys breadth, while the 'expert' training distribution conveys depth at critical regions of the support of the inference data distribution. For example, for a safety critical AI application like a self-driving car, a useful model should meld the massive amount of data gathered by everyday drivers under common conditions with the purposefully collected data gathered by automotive safety organizations using professional drivers on closed tracks under adverse conditions (or high-fidelity simulations thereof). Also, it should do so without compromising the privacy of everyday drivers.

In this vein, we can look at the mixing described in Section 2 as simplistic. It assumes that a simple *union* of two datasets (the decentralized training examples on edge devices and the centralized examples in a datacenter) can usefully match an inference distribution. But what if we need to be more intricate and selective in our use of e.g. centralized training data?

For example, our centralized corpus might be enormous, with most examples not germane to our immediate problem. Rather than using the corpus in its entirety, we'd like to filter to selectively use only the centralized data examples that are most relevant. E.g., we have access to a public corpus of photos, but wish to filter to just portraits. Alternatively, we might lack for any centralized corpus, and need to undertake a targeted data collection exercise. E.g. (along the lines of the scenario in Appendix B), all the federated data is smiling faces and we realize we need to organize the paid collection of portraits of unsmiling faces.

To undertake such filtering/targeting, we could use information from the federated private data examples (properly anonymized) as input. We wish to know enough about the federated training data distribution to understand what public data would be germane (e.g., portraits of human faces) but complementary (e.g., portraits with unsmiling subjects) to the decentralized data. Two tools for doing this, which could be combined with mixed learning, are federated analytics [Ramage and Mazzocchi, 2020] and federated private generative models [Augenstein et al., 2019].

It is also interesting to consider FL applications where privacy is not the dominant concern. Imagine the usage of FL for a multi-robot exploration application. Instead of privacy, bandwidth and communication latency between the robots and the coordinating server are the dominant constraints. The robots are priceless assets (e.g., Mars rovers), and the operations team must do everything possible to avoid catastrophic accidents that would end the robots' mission. As the robots collectively learn models in their environment, mixed learning can be used to augment (from long distance) the robots' locally acquired data with helpful information. Given the limited transmit capacity, the desire is to send only that information which best complements the data the robots already possess. The targeted selection, collection, or simulation of useful 'expert' centralized examples would be of utmost importance.

Furthermore, as the robots' onboard datasets would be continually growing and changing in the course of their explorations of new environments, the nature of the centralized information transmitted to the robots would also need to change over time. In this continual learning scenario, the mixing of these two training data distributions would call for *active* 'expert' centralized data collection.

In summary, the ability to measure and selectively choose which examples from a centralized dataset are most useful to aid a decentralized distribution in matching an inference distribution, and the ability to do this in a time-varying, 'online' manner, are key next steps to unlocking further privacy and utility wins via FL.

References

Apple. Designing for privacy (video and slide deck). Apple WWDC, `https://developer.apple.com/videos/play/wwdc2019/708`, 2019.

Sean Augenstein, H Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Trung Dang, Om Thakkar, Swaroop Indra Ramaswamy, Rajiv Mathews, Peter Chin, and Franccoise Beaufays. A method to reveal speaker identity in distributed asr training, and how to counter it. *ArXiv*, abs/2104.07815, 2021.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020a.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *International Conference on Machine Learning (ICML)*, 2020b.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282, 2017. Initial version posted on arXiv in February 2016.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*, 2018.

Daniel Ramage and Stefano Mazzocchi. Federated analytics: Collaborative data science without data collection. Google AI Blog, `https://ai.googleblog.com/2020/05/federated-analytics-collaborative-data.html`, 2020.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*, 2019.

TFF CelebA documentation. `tff.simulation.datasets.celeba.load_data` documentation, 2021. URL `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/celeba/load_data`.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Agüera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

White House Report. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *Journal of Privacy and Confidentiality*, 2013.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *ArXiv*, abs/1806.00582, 2018.

A Algorithms

Algorithms are provided for the mixing strategies described in Section 2. We first give the general FEDAVG algorithm (Algorithm 1). We then show how it is modified to mix in a centralized data distribution, via Synchronous Parallel Training (Algorithm 2), Example Transfer (Algorithm 3), or Gradient Transfer (Algorithm 4). The highlighted blue text indicates the steps added to basic FEDAVG in each case.

Algorithm 1: Generalized FEDAVG [McMahan et al., 2017, Wang et al., 2021]

Input: Initial model $\boldsymbol{x}^{(0)}$; CLIENTOPT, SERVEROPT with learning rate η, η_s

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Sample a subset $\mathcal{S}^{(t)}$ of clients
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Initialize local model $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Sample a batch $\mathcal{B}_i^{(k)}$ of client data
      Compute local stochastic gradient w.r.t. client batch $g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)})$
      Perform local update $\boldsymbol{x}_i^{(t,k+1)} = \text{CLIENTOPT}(\boldsymbol{x}_i^{(t,k)}, g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)}), \eta, t)$
    end
    Compute local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}_i^{(t,0)}$
  end
  Aggregate local changes $\Delta^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} p_i \Delta_i^{(t)} / \sum_{i \in \mathcal{S}^{(t)}} p_i$
  Update global model $\boldsymbol{x}^{(t+1)} = \text{SERVEROPT}(\boldsymbol{x}^{(t)}, -\Delta^{(t)}, \eta_s, t)$
end

Algorithm 2: Generalized FEDAVG in Parallel with Central Training

Input: Initial model $\boldsymbol{x}^{(0)}$; CLIENTOPT, SERVEROPT, CENTRALOPT, MERGEOPT with learning rate $\eta, \eta_s, \eta_c, \eta_m$

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Initialize central model $x_c^{(t,0)} = x^{(t)}$
  for step $j = 0, \ldots, J$ **do**
    Sample a batch $\mathcal{B}^{(j)}$ of centralized data
    Compute stochastic gradient w.r.t. centralized batch $g(\boldsymbol{x}_c^{(t,j)}; \mathcal{B}^{(j)})$
    Perform update $\boldsymbol{x}_c^{(t,j+1)} = \text{CENTRALOPT}(\boldsymbol{x}_c^{(t,j)}, g(\boldsymbol{x}_c^{(t,j)}; \mathcal{B}^{(j)}), \eta_c, t)$
  end
  Compute central model delta $\boldsymbol{\Delta}_c^{(t)} = \boldsymbol{x}_c^{(t,J)} - \boldsymbol{x}^{(t)}$
  Sample a subset $\mathcal{S}^{(t)}$ of clients
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Initialize local model $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Sample a batch $\mathcal{B}_i^{(k)}$ of client data
      Compute local stochastic gradient w.r.t. client batch $g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)})$
      Perform local update $\boldsymbol{x}_i^{(t,k+1)} = \text{CLIENTOPT}(\boldsymbol{x}_i^{(t,k)}, g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)}), \eta, t)$
    end
    Compute local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}_i^{(t,0)}$
  end
  Aggregate local changes $\Delta^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} p_i \Delta_i^{(t)} / \sum_{i \in \mathcal{S}^{(t)}} p_i$
  Compute federated model $\boldsymbol{x}_f^{(t)} = \text{SERVEROPT}(\boldsymbol{x}^{(t)}, -\Delta^{(t)}, \eta_s, t)$
  Compute federated model delta $\boldsymbol{\Delta}_f^{(t)} = \boldsymbol{x}_f^{(t)} - \boldsymbol{x}^{(t)}$
  Aggregate central model and federated model deltas $\boldsymbol{\Delta}^{(t)} = \alpha \boldsymbol{\Delta}_c^{(t)} + (1-\alpha) \boldsymbol{\Delta}_f^{(t)}$
  Update global model $\boldsymbol{x}^{(t+1)} = \text{MERGEOPT}(\boldsymbol{x}^{(t)}, -\boldsymbol{\Delta}^{(t)}, \eta_m)$
end

Algorithm 3: Generalized FEDAVG with Example Transfer

Input: Initial model $\boldsymbol{x}^{(0)}$; CLIENTOPT, SERVEROPT with learning rate η, η_s

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Sample a subset $\mathcal{S}^{(t)}$ of clients
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Sample a batch $\mathcal{B}^{(t)}$ of centralized data, send to client i
    Merge examples in $\mathcal{B}^{(t)}$ into existing client i data
    Initialize local model $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Sample a batch $\tilde{\mathcal{B}}_i^{(k)}$ of mixed data
      Compute local stochastic gradient w.r.t. mixed batch $g_i(\boldsymbol{x}_i^{(t,k)}; \tilde{\mathcal{B}}_i^{(k)})$
      Perform local update $\boldsymbol{x}_i^{(t,k+1)} = \text{CLIENTOPT}(\boldsymbol{x}_i^{(t,k)}, g_i(\boldsymbol{x}_i^{(t,k)}; \tilde{\mathcal{B}}_i^{(k)}), \eta, t)$
    end
    Compute local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}_i^{(t,0)}$
  end
  Aggregate local changes $\Delta^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} p_i \Delta_i^{(t)} / \sum_{i \in \mathcal{S}^{(t)}} p_i$
  Update global model $\boldsymbol{x}^{(t+1)} = \text{SERVEROPT}(\boldsymbol{x}^{(t)}, -\Delta^{(t)}, \eta_s, t)$
end

Algorithm 4: Generalized FEDAVG with Gradient Transfer

Input: Initial model $\boldsymbol{x}^{(0)}$; CLIENTOPT, SERVEROPT with learning rate η, η_s

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Sample a batch $\mathcal{B}^{(t)}$ of centralized data
  Compute stochastic gradient w.r.t. centralized batch $g(\boldsymbol{x}^{(t)}; \mathcal{B}^{(t)})$
  Sample a subset $\mathcal{S}^{(t)}$ of clients
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Initialize local model $\boldsymbol{x}_i^{(t,0)} = \boldsymbol{x}^{(t)}$
    Set augmenting gradient $\tilde{g}^{(t)} = g(\boldsymbol{x}^{(t)}; \mathcal{B}^{(t)})$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Sample a batch $\mathcal{B}_i^{(k)}$ of client data
      Compute local stochastic gradient w.r.t. client batch $g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)})$
      Perform local update $\boldsymbol{x}_i^{(t,k+1)} = \text{CLIENTOPT}(\boldsymbol{x}_i^{(t,k)}, g_i(\boldsymbol{x}_i^{(t,k)}; \mathcal{B}_i^{(k)}) + \tilde{g}^{(t)}, \eta, t)$
    end
    Compute local model changes $\Delta_i^{(t)} = \boldsymbol{x}_i^{(t,\tau_i)} - \boldsymbol{x}_i^{(t,0)}$
  end
  Aggregate local changes $\Delta^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} p_i \Delta_i^{(t)} / \sum_{i \in \mathcal{S}^{(t)}} p_i$
  Update global model $\boldsymbol{x}^{(t+1)} = \text{SERVEROPT}(\boldsymbol{x}^{(t)}, -\Delta^{(t)}, \eta_s, t)$
end

B Experiment

We present an experiment which explores the strategies described in Section 2.

Scenario Consider a mobile phone camera portrait application, with a feature desired that would identify if human subjects in photos are smiling (e.g., to suggest to users if a portrait photo should be kept or needs retaking). In the FL paradigm, models are trained on actual user images on device (under stringent user privacy protections, e.g. the images never leave the device). However, as the application usage is one that generally results in only positive (i.e., smiling) example portraits being retained by users, training a binary 'smiling'-vs.-'unsmiling' classifier from only the edge device data

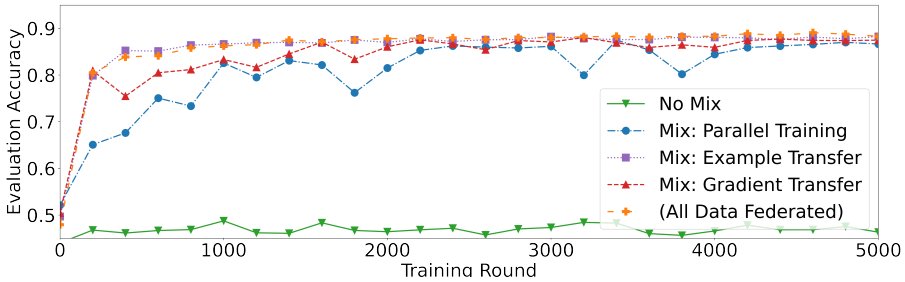

Figure 1: CelebA smiling classifier, evaluation accuracy vs. federated training round.

will be challenging due to label imbalance. We can leverage the fact that many large public corpora of photos of human faces exist, containing many images of unsmiling subjects. We examine mixing this public data into our FL training process, using the strategies presented in Section 2.

Datasets We use a version of the CelebA dataset [Liu et al., 2015], made into a federated dataset[1] by partitioning images up by portrait subject [Caldas et al., 2018]. I.e., we treat each celebrity as having a mobile phone with a cache of photos of themselves, which participates as a client in FL. The CelebA data contains a boolean attribute indicating whether a portrait subject is smiling or not. We filter on this attribute, to form two resulting groups of data: a federated dataset partitioned by celebrity, containing only smiling images, and a centralized dataset containing all unsmiling images together (removing partitions and putting all celebrities together in a single set). The former represents the on-device user private data, and the latter is a publicly available corpus of unsmiling faces.

Setup We use the federated and centralized datasets to learn a smiling classifier, under five scenarios:

i No Mixing, pure FEDAVG (Algorithm 1) (no access to any unsmiling examples)

ii Mixing via Parallel Training (Algorithm 2)

iii Mixing via Example Transfer (Algorithm 3)

iv Mixing via Gradient Transfer (Algorithm 4)

v No Mixing, pure FEDAVG, but with all data federated, both smiling and unsmiling examples (i.e., as if celebrities also retained unsmiling portraits on their mobile devices)

Scenario i is the (futile) case of training a binary classifier with no access to negative examples. Scenario v provides 'oracle' performance; it violates the fundamental assumption that negative examples are not present on user devices, but shows what pure FL performance would look like were this not the case. Scenarios ii-iv each cover a mixing strategy described in Section 2.

Of the 9343 total clients in federated CelebA, we train on a population of 8408 clients (approximately 90% of the total), and then evaluate on a separate population of the 935 remaining clients. We train for 5000 federated rounds, randomly selecting 100 training clients per round.

Our evaluation data consists of both smiling and unsmiling faces, and is meant to stand in for the inference distribution (where accurate classification of both smiling and unsmiling inputs is necessary). Note that CelebA contains smiling and unsmiling faces in nearly equal amounts, so a high evaluation accuracy cannot come at the expense of one particular label being poorly classified.

Results Figure 1 shows the classifier evaluation accuracy resulting from each mixing strategy. The three mixing strategies all result in a smiling classifier that is far more accurate as compared to the 'no mixing' scenario. See discussion in Section 3.

[1]The CelebA federated dataset is available via open source FL software [TFF CelebA documentation, 2021].