arXiv:1909.06335v1 [cs.LG] 13 Sep 2019

Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification

Tzu-Ming Harry Hsu[*]
MIT CSAIL
stmharry@mit.edu

Hang Qi
Google Research
hangqi@google.com

Matthew Brown
Google Research
mtbr@google.com

Abstract

Federated Learning enables visual models to be trained in a privacy-preserving way using real-world data from mobile devices. Given their distributed nature, the statistics of the data across these devices is likely to differ significantly. In this work, we look at the effect such non-identical data distributions has on visual classification via Federated Learning. We propose a way to synthesize datasets with a continuous range of identicalness and provide performance measures for the Federated Averaging algorithm. We show that performance degrades as distributions differ more, and propose a mitigation strategy via server momentum. Experiments on CIFAR-10 demonstrate improved classification performance over a range of non-identicalness, with classification accuracy improved from 30.1% to 76.9% in the most skewed settings.

server momentum
for tackling
heterogenity.

1 Introduction

Federated Learning (FL) [McMahan et al., 2017] is a privacy-preserving framework for training models from decentralized user data residing on devices at the edge. With the Federated Averaging algorithm (`FedAvg`), in each federated learning round, every participating device (also called *client*), receives an initial model from a central server, performs stochastic gradient descent (SGD) on its local dataset and sends back the gradients. The server then aggregates all gradients from the participating clients and updates the starting model. Whilst in data-center training, batches can typically be assumed to be IID (independent and identically distributed), this assumption is unlikely to hold in Federated Learning settings. In this work, we specifically study the effects of non-identical data distributions at each client, assuming the data are drawn independently from differing local distributions. We consider a continuous range of non-identical distributions, and provide empirical results over a range of hyperparameters and optimization strategies.

study effect
of non-iid
behavior.

2 Related Work

Several authors have explored the `FedAvg` algorithm on non-identical client data partitions generated from image classification datasets. McMahan et al. [2017] synthesize pathological non-identical user splits from the MNIST dataset, sorting training examples by class labels and partitioning into shards such that each client is assigned with 2 shards. They demonstrate that `FedAvg` on non-identical clients still converges to 99% accuracy, though taking more rounds than identical clients. In a similar sort-and-partition manner, Zhao et al. [2018] and Sattler et al. [2019] generate extreme partitions on the CIFAR-10 dataset, forming a population consisting of 10 clients in total. These settings are somewhat unrealistic, as practical federated learning would typically involve a larger pool of clients, and more complex distributions than simple partitions.

[*] work done while interning at Google

Preprint. Under review.

Other authors look at more realistic data distributions at the client. For example, Caldas et al. [2018] use Extended MNIST [Cohen et al., 2017] with partitions over writers of the digits, rather than simply partitioning over digit class. Closely related to our work, Yurochkin et al. [2019] use a Dirichlet distribution with concentration parameter 0.5 to synthesize non-identical datasets. We extend this idea, exploring a continuous range of concentrations α, with a detailed exploration of optimal hyperparameter and optimization settings.

EMNIST is good for FL !

Prior work on the theoretical side studied the convergence of FedAvg variants under different conditions. Sahu et al. [2018] introduce a proximal term to client objectives and prove convergence guarantees. Li et al. [2019] analyze FedAvg under proper sampling and averaging schemes in strongly convex problems.

3 Synthetic Non-Identical Client Data

In our visual classification task, we assume on every client training examples are drawn independently with class labels following a categorical distribution over N classes parameterized by a vector $\boldsymbol{q}$ ($q_i \geq 0, i \in [1, N]$ and $\|\boldsymbol{q}\|_1 = 1$). To synthesize a population of non-identical clients, we draw $\boldsymbol{q} \sim \text{Dir}\,(\alpha \boldsymbol{p})$ from a Dirichlet distribution, where $\boldsymbol{p}$ characterizes a prior class distribution over N classes, and $\alpha > 0$ is a *concentration* parameter controlling the identicalness among clients. We experiment with 8 values for α to generate populations that cover a spectrum of identicalness. With $\alpha \to \infty$, all clients have identical distributions to the prior; with $\alpha \to 0$, on the other extreme, each client holds examples from only one class chosen at random.

In this work, we use the CIFAR-10 [Krizhevsky et al., 2009] image classification dataset, which contains 60,000 images (50,000 for training, 10,000 for testing) from 10 classes. We generate balanced populations consisting of 100 clients, each holding 500 images. We set the prior distribution to be uniform across 10 classes, identical to the test set on which we report performance. For every client, given an α, we sample $\boldsymbol{q}$ and assign the client with the corresponding number of images from 10 classes. Figure 1 illustrates populations drawn from the Dirichlet distribution with different concentration parameters.

Nice & intuitive visualization.

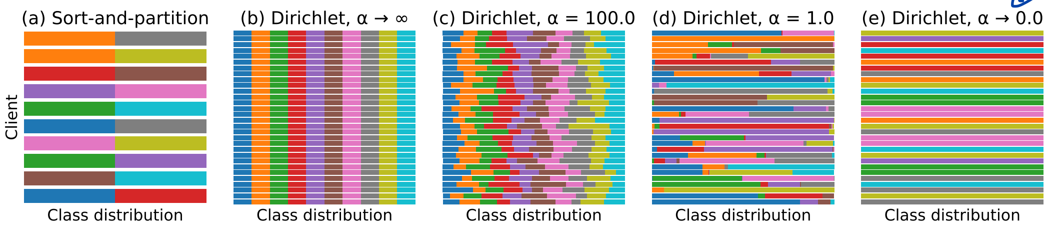

Figure 1: **Synthetic populations with non-identical clients.** Distribution among classes is represented with different colors. (a) 10 clients generated from the sort-and-partition scheme, each assigned with 2 classes. (b–e) populations generated from Dirichlet distribution with different concentration parameters α respectively, 30 random clients each.

Common!

4 Experiments and Results

Given the above dataset preparation, we now proceed to benchmark the performance of the vanilla FedAvg algorithm across a range of distributions ranging from identical to non-identical.

We use the same CNN architecture and notations as in McMahan et al. [2017] except that a weight decay of 0.004 is used and *no* learning rate decay schedule is applied. This model is not the state-of-the-art on the CIFAR-10 dataset, but is sufficient to show relative performance for the purposes of our investigation.

don't need to use SOTA models when showing relative performance.

FedAvg is run under client batch size $B = 64$, local epoch counts $E \in \{1, 5\}$, and reporting fraction $C \in \{0.05, 0.1, 0.2, 0.4\}$ (corresponding to 5, 10, 20, and 40 clients participating in every single round, respectively) for a total of 10,000 communication rounds. We perform hyperparameter search over a grid of client learning rates $\eta \in \left\{10^{-4}, 3 \times 10^{-4}, \ldots, 10^{-1}, 3 \times 10^{-1}\right\}$.

4.1 Classification Performance with Non-Identical Distributions

Figure 2 shows classification performance as a function of the Dirichlet concentration parameter α (larger α implies more identical distributions). Significant changes in test accuracy occur around low α when the clients come close to one-class. Increasing the reporting fraction C yields diminishing returns, and the gain in performance is especially marginal for identically distributed client datasets. Interestingly, for the case of fixed optimization round budget, synchronizing the weights more frequently ($E = 1$) does not always improve the accuracy on non-identical data.

fraction of sampled clients.

In addition to reduced end-of-training accuracy, we also observe more volatile training error in the case of more non-identical data, see Figure 3. Runs with small reporting fraction struggle to converge within 10,000 communication rounds.

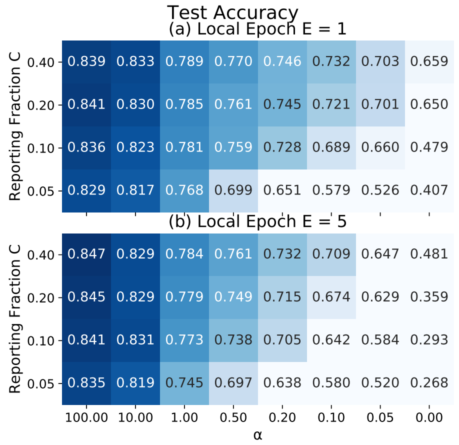

Figure 2: **FedAvg accuracy for different α.** Each cell is optimized over learning rates, with each learning rate averaged over 5 runs on different populations under the same α.

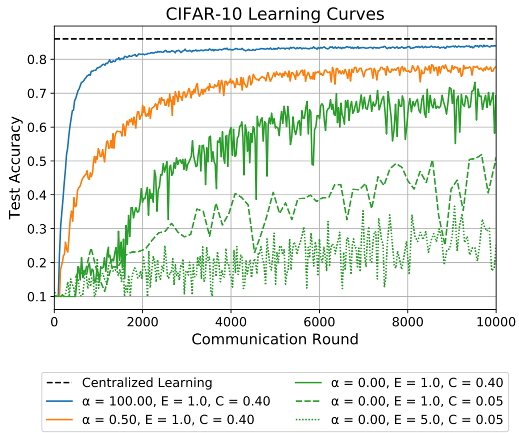

Figure 3: **FedAvg learning curves with fixed learning rates.** The centralized learning result (dashed line) is from TensorFlow tutorial [TensorFlow].

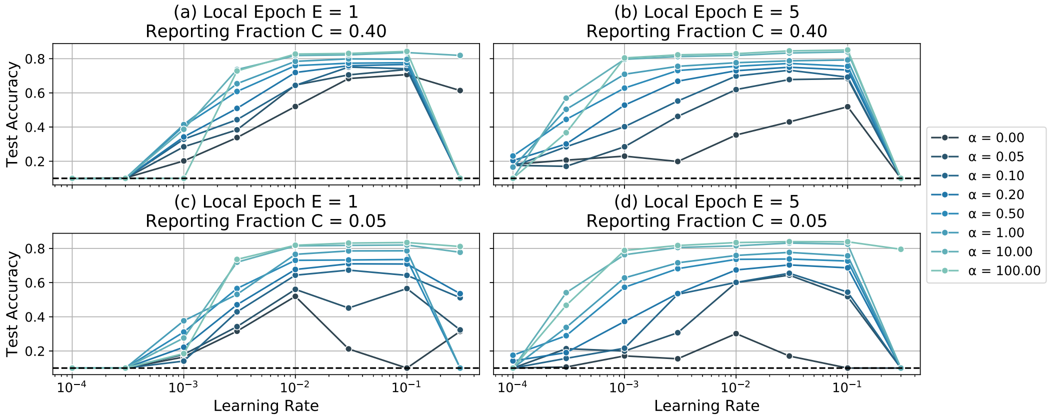

Figure 4: **FedAvg test accuracy in hyperparameter search.** (a–b) High and (c–d) low reporting fraction out of 100 clients are demonstrated. Chance accuracy is shown by the dashed line.

Hyperparameter sensitivity. As well as affecting overall accuracy on the test set, the learning conditions as specified by C and α have a significant effect on hyperparameter sensitivity. On the identical end with large α, a range of learning rates (about two orders of magnitude) can produce good accuracy on the test set. However, with smaller values of C and α, careful tuning of the learning rate is required to reach good accuracy. See Figure 4.

4.2 Accumulating Model Updates with Momentum

Using momentum on top of SGD has proven to have great success in accelerating network training by a running accumulation of the gradient history to dampen oscillations. This seems particularly relevant for FL where participating parties may have a sparse distribution of data, and hold a limited subset of labels. In this subsection we test the effect of momentum at the server on the performance of `FedAvg`.

Vanilla `FedAvg` updates the weights via $\boldsymbol{w} \leftarrow \boldsymbol{w} - \Delta\boldsymbol{w}$, where $\Delta\boldsymbol{w} = \sum_{k=1}^{K} \frac{n_k}{n} \Delta\boldsymbol{w}_k$ (n_k is the number of examples, $\Delta\boldsymbol{w}_k$ is the weight update from k'th client, and $n = \sum_{k=1}^{K} n_k$). To add momentum at the server, we instead compute $\boldsymbol{v} \leftarrow \beta\boldsymbol{v} + \Delta\boldsymbol{w}$, and update the model with $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{v}$. We term this approach `FedAvgM` (Federated Averaging with Server Momentum).

FedAvg with Server Momentum!

In experiments, we use Nesterov accelerated gradient [Nesterov, 2007] with momentum $\beta \in \{0, 0.7, 0.9, 0.97, 0.99, 0.997\}$. The model architecture, client batch size B, and learning rate η are the same as vanilla `FedAvg` in the previous subsection. The learning rate of the server optimizer is held constant at 1.0.

Effect of server momentum. Figure 5 shows the effect of learning with non-identical data both with and without server momentum. The test accuracy improves consistently for `FedAvgM` over `FedAvg`, with performance close to the centralized learning baseline (86.0%) in many cases. For example, with $E = 1$ and $C = 0.05$, `FedAvgM` performance stays relatively constant and above 75%, whereas `FedAvg` accuracy falls rapidly to around 35%.

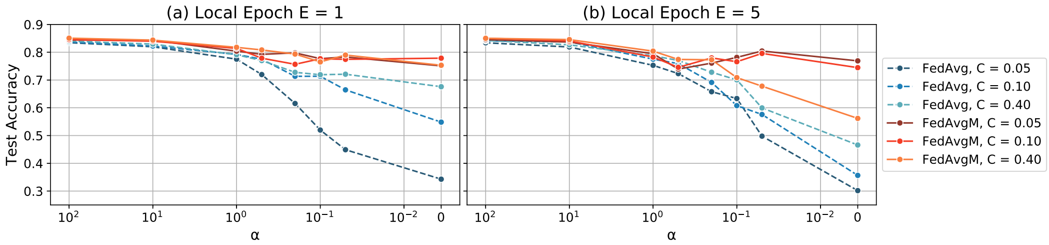

Figure 5: **FedAvgM and FedAvg performance curves for different non-identical-ness.** Data is increasingly non-identical to the right. Best viewed in color.

Hyperparameter dependence on C and E. Hyperparameter tuning is harder for `FedAvgM` as it involves an additional hyperparameter β. In Figure 6, we plot the accuracy against the effective learning rate defined as $\eta_{\text{eff}} = \eta/(1-\beta)$ [Shallue et al., 2018] which suggests an optimal η_{eff} for each set of learning conditions. Notably, when the reporting fraction C is large, the selection of η_{eff} is easier and a range of values across two orders of magnitude yields reasonable test accuracy. In contrast, when only a few clients are reporting each round, the viable window for η_{eff} can be as small as just one order of magnitude. To prevent client updates from diverging, we additionally have to use a combination of low absolute learning rate and high momentum. The local epoch parameter E affects the choice of learning rate as well. Extensive local optimization increases the variance of clients' weight updates, therefore a lower η_{eff} is necessary to counteract the noise.

use effec. LR $\eta_{eff.} = \eta/(1-\beta).$

more local optimizat. increases variance of client's weight updates.

lower η_{eff} needed to counter noise

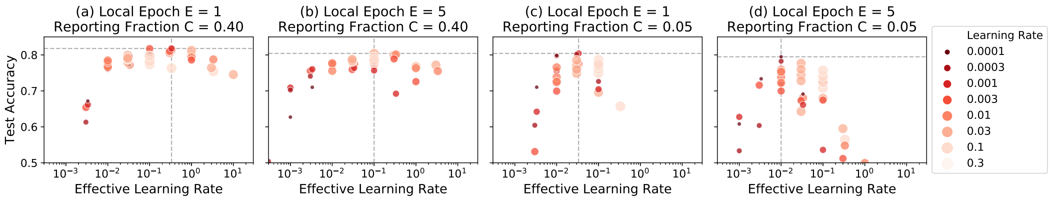

Figure 6: **Sensitivity of test accuracy for FedAvgM.** Plotted for $\alpha = 1$. The effective learning rate is defined as $\eta_{\text{eff}} = \eta/(1-\beta)$. Sizes are proportional to client learning rate η and the most performant point is marked by crosshair.

References

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

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-IID data. *arXiv preprint arXiv:1907.02189*, 2019.

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*, pages 1273–1282, 2017.

Yu Nesterov. Gradient methods for minimizing composite objective function. 2007.

Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-IID data. *arXiv preprint arXiv:1903.02891*, 2019.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

TensorFlow. Advanced convolutional neural networks. URL https://www.tensorflow.org/tutorials/images/deep_cnn.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261, 2019.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-IID data. *arXiv preprint arXiv:1806.00582*, 2018.