Accuracy Tradeoffs of Federated Learning approaches

Ilia Khitrov, Christopher Harth-Kitzerow*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ilia.khitrov@tum.de, christopher.harth-kitzerow@net.in.tum.de

***Abstract*—Federated Learning is a technique that implements distributed privacy preserving machine learning principles. The idea is useful for scenarios where data owners do not want to reveal private data, but a model that takes into account multiple databases is desired. However, a trade off between accuracy and privacy is implied for such approaches. We review well known algorithms of federated learning including Federated averaging, FedSGD, FedProx, FedNOVA, SCAFFOLD and SecureBoost, and compare their accuracy with each other and with a minibatch baseline. Ranging scenarios for applications of Federated Learning algorithms are discussed and existing theoretical advantages and limitations are mentioned.**

***Index Terms*—federated learning, benchmarks, accuracy**

1. Introduction

We live in a data driven world where every day enormous amounts of data are collected directly from our mobile devices and personal computers. Analyzing this data is a rapidly developing area both in research and application fields widely known as machine learning (ML). The most common techniques in ML today rely on centralized approaches, which impose problems with privacy in case of data leakage and can require powerful computation units [1]. Federated Learning (FL) is an approach that attempts to solve both of these problems by using distributed computations and encrypted information exchange between these units. In recent years, a lot of research was dedicated to developing algorithms for privacy-preserving machine learning, and FL has been proved to be a reliable and efficient way to deal with such challenges [2].

In this paper we will review the existing approaches for Federated Learning, key ideas behind them, and scenarios in which they are the most effective, as well as compare their performance with models trained on centralized data. We will start by introducing to the readers the ideas behind FL approaches and categorization of such algorithms.

2. Overview of Federated Learning

2.1. Definition of FL and notation

In a typical application scenario [1], the goal of Federated Learning algorithm is to train a model by collecting training information from distributed devices without revealing actual training datasets to the organizer. Let K be the set of data owners (clients), indexed as k, with corresponding datasets $D_1, ...D_k$; none of data owners has direct access to other clients' data. The algorithm consists of the following basic steps [1] [2]:

1) Clients for next training iteration are selected by the server;
2) Current machine learning model W is communicated to selected clients;
3) Each client k keeps their local databases D_k private and uses them to update weights of individual models W_k;
4) Server collects local models $W_1, ...W_k$ and aggregates them to update the global model W'.

The algorithm is visualized in a typical application scenario by Google in Figure 1. Scenarios may differ from the one mentioned here, e.g. direct communication and aggregation of updated weights is possible [3]. However, such approaches lie beyond the scope of this paper.

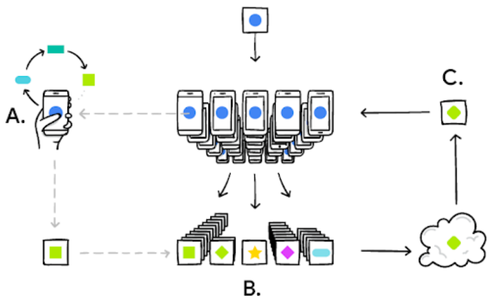


Figure 1: Each client's phone personalizes the model locally, based on their usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated. [4]

2.2. Categorizations of FL Algorithms

Federated learning algorithms can be classified by type of data partitioning, used machine learning model, privacy mechanism, and communications architecture. [2]. Since this paper is focused on accuracy trade off only first two classifications are discussed here.

2.2.1. Data partitioning.

Another study proposed categorization of tasks for FL algorithms which is based on data split pattern [3].

Horizontal federated learning (a), also known as sample-based federated learning, is applicable in scenarios where all individual datasets D_k have about the same feature space but are different in samples. FL systems aimed for such scenarios usually have specific architecture in which all clients compute local gradient and communicate it to the server for aggregation into global model. One of the major complications here is that real world datasets are rarely independent and identically distributed (IID), which causes problems for aggregation of local models [1].

Vertical federated learning (b) is applied when feature spaces of datasets D_k overlap only a little, but the sets of IDs overlap significantly. In this scenario various ML algorithms proved to be effective, including statistical analysis, gradient descent and safe linear regression [2].

Federated transfer learning (c) is applied to scenarios when neither feature spaces nor IDs spaces of datasets are close to each other [1]. Such algorithms require complicated architectures and are not reviewed in this paper.

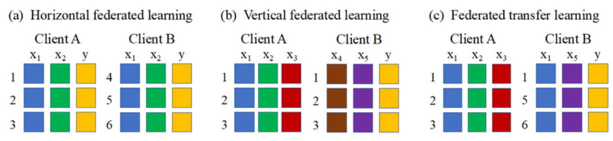

(a) Horizontal federated learning
(b) Vertical federated learning
(c) Federated transfer learning
Client A
Client B
x1 x2 y
x1 x2 x3
x4 x5 y
1
2
3
4
5
6

Figure 2: Categorization of Federated Learning by data partitioning. [5]

2.2.2. Applicable Machine Learning model.

Three families of ML algorithms to be combined with federated learning approaches can be considered: linear models, tree models, neural networks (NNs) [2]. Each type of ML model can be useful under constrains on performance of clients' devices, data types and distribution of data between clients. However, NNs and some tree boosting algorithms are more popular today [3].

3. Original Federated Learning Algorithms

3.1. Background

In 2017, McMahan et al. introduced the term Federated Learning and created two implementations that formed the basis of the whole branch. Both of the approaches use stochastic gradient descent (SGD), as most of successful applications of deep learning relied on this technique at the time [6]. Minibatch SGD is used as a reference (and baseline for benchmarks) for these algorithms, since it is controlled by hyperparameters that can be easily adopted for distributed ML techniques. The paper proposed that a C-fraction of clients is selected on each round t, and gradient of the loss function over the D_k is computed. Selected clients performs E epochs of local-update SGD with a mini-batch size B; loss function $l(w, D_k)$ is to be minimized. P_k denotes the index set of samples in D_k, n being the total number of samples. This approach is focused on non-convex neural networks objectives, but the architecture can be applied to wider family of ML models. These algorithms are widely used in scenarios of horizontal federated learning [1].

3.2. The algorithms

3.2.1. FedSGD.

The Federated Stochastic Gradient Descent (FedSGD) algorithm derives its settings from large-batch synchronous SGD. This algorithm is referred to as a naive approach for the FL algorithm. Proposed implementation with $C = 1$ (i. e. full-batch gradient descent) and fixed learning rate η includes each client k computing averaged gradient on D_k: $g_k = \nabla F_k(w_t)$, w_t being the current model, and then central server updates w_t using aggregated gradients: $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$.

3.2.2. FedAvg.

For Federated Averaging (FedAvg) algorithm, more computation is added to each client by iterating the local update multiple times before averaging. Algorithm 1 is an exact reproduction of pseudocode from [1]. Three parameters are controlling the amount of computation and accuracy of the final model: selected fraction of clients C, number of epochs E each data owner goes through his batch to compute local gradient, and local minibatch size $B (B = \infty$ indicates usage of the whole local dataset as training dataset); the algorithm with $B = \infty$ and $C = 1$ is the same as FedSGD described earlier.

Algorithm 1 Federated Average, [6]

```
Server executes:
initialize $w_0$
for each round $t = 1, 2, ...$ do
    $m \leftarrow max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ items)
    for each client $k \in S_t$ in parallel do
        $w_{t+1}^k \leftarrow$ ClientUpdate $(k, w_k)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

ClientUpdate$(k, w)$:
$\mathcal{B} \leftarrow$ (split $P_k$ into batches of size $B$)
for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
        $w \leftarrow w - \eta \nabla l(w, b)$
    end for
    return $w$ to server
end for
```

4. Further Improvements

Since 2017, when FedAvg was introduced, many attempts to improve this algorithm were made [1].Some of the most established approaches are mentioned in this chapter.

4.1. FedProx

A detailed review of theFederated averaging technique pointed out the importance and influence of number of epochs E on the resulting performance of the algorithm [7]. From obtaining these insights a modification of this algorithm called FedProx was introduced. In this approach each client minimizes approximated loss function instead of exact one. In particular, local updated weights w_{t+1}^k are obtained by solving $\min_w h_k(w, w_t) = l(w, b) + \frac{\mu}{2}||w - w_t||^2$, i.e. minimizing loss function l with given batch b so that $||w - w_t||^2 < \epsilon$. This is a simple way to ensure that local updates are not too far from the global optima. μ is another hyperparameter that has to be tuned; very low values of μ impose almost no regularization effect, and large values cause a drastic slowdown of convergence rate.

4.2. SCAFFOLD

Another study introduced the algorithm SCAFFOLD (Stochastic controlled averaging for federated learning) which aimed to improve performance of Federated averaging in case of non-IID partitioned dataset by using variance reduction techniques [8]. Control variates are introduced both for the server (c) and clients (c_k), and are used to determine corresponding update directions. These variates are updated either by reuse of previously calculated gradients or by computing values of gradients of the local datasets on global model. First approach implements idea similar to Gradient descent with momentum and has lower computation costs, while the second tends to be more stable. This algorithm can significantly improve convergence rate, but communication costs are double the costs of FedAvg because of additional control variates.

4.3. FedNOVA

The Federated normalized averaging (FedNOVA) algorithm is also similar to FedAvg, but has improved aggregation stage. In this algorithm the local updates w_{t+1}^k are scaled in accordance with the actual volume of locally performed training, which may be different because of time constraints combined with different computation speeds of clients, or ranging sizes of local datasets. This allows prioritizing better-trained local weights while keeping communication costs at the same level as FedAvg and only slightly enlarging local computation costs [9].

4.4. Worth noting

Besides algorithms reviewed here, a lot of different federated learning approaches were introduced. Out of this large number of algorithms we considered worth noting the FedPAGE [10], which improves convergence rate in both convex and non-convex settings by implementing recent probabilistic gradient estimator instead of SGD and FedFA [11], which aims at achieving better accuracy and fairness in horizontal federated learning scenarios by employing double momentum gradient and new weight selection algorithm. We consider analyzing accuracy of these and other algorithms as a possibility for future work.

5. Accuracy Analysis of Horizontal Federated Learning Approaches

5.1. FedAvg and FedSGD

5.1.1. Synthetic IID data. Experiments showed that FedSGD and FedAvg can both perform well with a wide range of ML models including multi-layer perceptron, convolutional NNs, two-layer character Long short-term memory (LSTM) networks and large-scale word-level LSTM, and FedAvg tends to reach better test accuracy than FedSGD with same number of communication rounds. To give a better insight of how these FL approaches perform, let us review benchmarks performed on CIFAR-10 [12] with balanced and IID data partitioning [6]. It is shown that FedAvg reaches an accuracy of 85% already after 2000 communication rounds, while centralized SGD needs around 197500 communications to show a similar accuracy of 86%, since communication after every batch is assumed. However, if compared by number of minibatch computations, SGD on united data has better convergence, as gradient updates after each minibatch computation (see Figure 3).

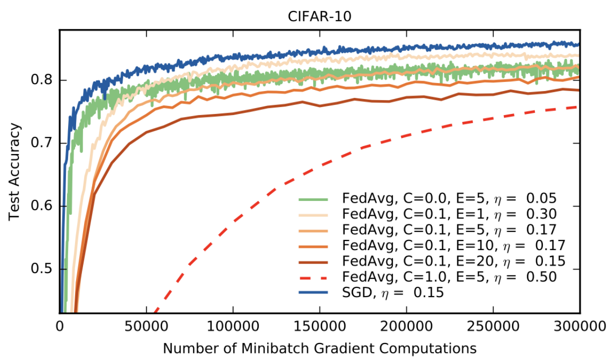


Figure 3: Comparison of accuracy on IID data [6].

5.1.2. Further evaluation. In a later study [13] benchmark aimed specifically for testing performance from different perspectives was developed. It was shown that both these FL approaches converged towards an accuracy level similar to that of a centralized ML model, and significantly higher than that of the same ML model trained only on local data; these results were obtained on IID partitioned MNIST, FEMNIST and ColabA datasets using MLP and LeNet ML models. For robustness evaluation the same datasets were used, but now with non-IID partitioning between clients; it was shown that accuracy of FL approaches drops when clients have only instances of few classes in their training datasets, e. g. in the scenario when training dataset was split so that each client has only one class of samples (see Figure 4).

Theoretical analysis showed [14] that for quadratic objectives performance of FL approaches is strictly better then that of minibatch algorithms, and accelerated variant is minimax optimal. For general convex objectives it is proven that first error upper bound of FL is not worse than that of minibatch SGD, if typical noise scaling is applied. However, lower bound of the error of local SGD

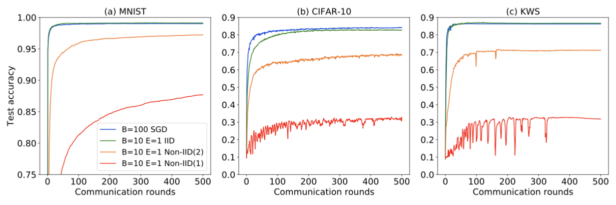
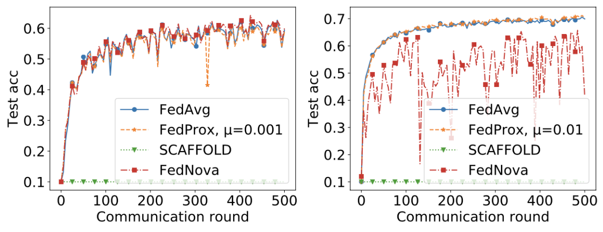


Figure 4: Comparison of accuracy on non-IID data. [13]

approach in worst-case scenario is higher than the worst-case error of minibatch SGD.

5.2. Other Algorithms

Performance of FedAvg, FedNOVA, SCAFFOLD and FedProx was analyzed in the setting of non-IID partitioned dataset [15]. This study showed that none of mentioned algorithms dominates others, and each of them can show outstanding performance under specific constrains. In case of label distribution skew or quantity skew FedProx usually performs better than mentioned alternatives. The accuracy of SCAFFOLD is quite unstable, but in some cases it can significantly outperform other approaches.FedNOVA does not show superiority inn analyzed scenarios, but can sometimes be slightly more effective than other approaches. To illustrate behavior of different approaches, their learning curves on CIFAR-10 [12] with partition in accordance with Dirichlet distribution are shown in Figure 5. We chose this dataset as it was stated to be a challenging task for federated learning approaches under non-IID conditions.



(a) $p_k \sim Dir(0.5)$ (b) $q \sim Dir(0.5)$

Figure 5: Comparison of accuracy on non-IID data from CIFAR-10 with 100 parties, $C = 0.1$, Dirichlet distribution is used to simulate Label distribution skew (left) and Quantity skew (right) [15].

6. Approaches for Vertical Federated Learning

6.1. SecureBoost

A recent study introduced a new algorithm called SecureBoost that is aimed at vertical federated learning [16]. This system implements federated tree-boosting ML model in accordance with principles of FL that is lossless, i.e. is as accurate as other non-federated tree-boosting algorithms trained on centralized data. SecureBoost is based on the XGBoost algorithm, which was recently shown to be one of the most effective ways to work with panel data [17].

6.2. SecureBoost+

Slightly modified version of this algorithm with improvements in performance on large and high dimensional datasets called SecurityBoost+ was recently published [18]. The new approach converges towards similar accuracy, but can be trained much faster to reach the same classification error. The following table illustrates their performance.

Data set	XGB	SecureBoost	SecureBoost+
Give-credit	0.872	0.874	0.873
Susy	0.864	0.873	0.873
Higgs	0.808	0.806	0.8
Epsilon	0.897	0.897	0.894

Figure 6: Area under the receiver operating characteristic (ROC) curve for SecureBoost, SecureBoost+ and XGBoost on centralized data for different datasets [18].

7. Conclusion

In this paper we surveyed the literature on various implementations of federated learning approaches and provided information on existing benchmarks of these algorithms, analyzing their accuracy in different scenarios. We showed that existing implementations can demonstrate very similar accuracy in comparison with centralized approaches while having advantages in terms of privacy, and are guaranteed to reach the same accuracy in some scenarios. However, some settings can impose difficulties for such approaches, e.g. non-IID partitioned datasets held by clients, which is quite common in real-world problems. Depending on the scenario different approaches reviewed in this article can show better performance, and none of them can be considered as universally best one.

References

[1] H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1, 2021.

[2] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[4] "Federated learning: Collaborative machine learning without centralized training data," Apr 2017. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[5] S. Chen, D. Xue, G. Chuai, Q. Yang, and Q. Liu, "Fl-qsar: a federated learning based qsar prototype for collaborative drug discovery," 02 2020.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[7] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 3, p. 3, 2018.

[8] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, “Scaffold: Stochastic controlled averaging for federated learning,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[9] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[10] H. Zhao, Z. Li, and P. Richtárik, “Fedpage: A fast local stochastic gradient method for communication-efficient federated learning,” *arXiv preprint arXiv:2108.04755*, 2021.

[11] W. Huang, T. Li, D. Wang, S. Du, and J. Zhang, “Fairness and accuracy in federated learning,” *arXiv preprint arXiv:2012.10069*, 2020.

[12] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[13] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” 2018. [Online]. Available: https://arxiv.org/abs/1806.00582

[14] B. Woodworth, K. K. Patel, S. Stich, Z. Dai, B. Bullins, B. Mcmahan, O. Shamir, and N. Srebro, “Is local sgd better than minibatch sgd?” in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 334–10 343.

[15] Q. Li, Y. Diao, Q. Chen, and B. He, “Federated learning on non-iid data silos: An experimental study,” *arXiv preprint arXiv:2102.02079*, 2021.

[16] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, “Secureboost: A lossless federated learning framework,” *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[17] R. Shwartz-Ziv and A. Armon, “Tabular data: Deep learning is not all you need,” *Information Fusion*, vol. 81, pp. 84–90, 2022.

[18] W. Chen, G. Ma, T. Fan, Y. Kang, Q. Xu, and Q. Yang, “Secureboost+: A high performance gradient boosting tree framework for large scale vertical federated learning,” *arXiv preprint arXiv:2110.10927*, 2021.