LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical data

Li Huang\(^2\), Yifeng Yin\(^3\), Zeng Fu\(^4\), Shifa Zhang\(^1,5\), Hao Deng\(^6\), and Dianbo Liu\(^1,7,\ast\)

\(^1\)Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, USA, MA 02139
\(^2\)The Future Lab and Academy of Arts and Design, Tsinghua University, Beijing, China, 10084
\(^3\)University of Huddersfield, Huddersfield, UK, HD1 3DH
\(^4\)University of California San Diego, San Diego, USA, CA 92093
\(^5\)Northeastern University, Boston, USA, MA 02115
\(^6\)Department of Anesthesia, Massachusetts General Hospital, Boston, USA, MA 02114
\(^7\)Harvard Medical School and Boston Children’s Hospital, Boston, USA, MA 02215
\(^\ast\)corresponding author: dianbo@mit.edu

ABSTRACT

Medical data are valuable for improvement of health care, policy making and many other purposes. Vast amount of medical data are stored in different locations, on many different devices and in different data silos. Sharing medical data among different sources is a big challenge due to regulatory, operational and security reasons. One potential solution is federated machine learning, which is a method that sends machine learning algorithms simultaneously to all data sources, train models in each source and aggregates the learned models. This strategy allows utilization of valuable data without moving them. One challenge in applying federated machine learning is the heterogeneity of data from different sources. To tackle this problem, we proposed an adaptive boosting method that increases the efficiency of federated machine learning. Using intensive care unit data from hospital, we showed that LoAdaBoost federated learning outperformed baseline method and increased communication efficiency at negligible additional cost.

Introduction

Health data can be used by medical practitioners to provide health care and by researchers to build machine learning models to improve clinical services and make health predictions. But such data is mostly stored distributively on mobile devices or in different hospitals because of its large volume and high privacy, implying that traditional learning approaches on centralized data may not be viable. Therefore, federated learning that avoids data collection and central storage becomes necessary and up to now significant progress has been made.

In 2005, Rehak \textit{et al.}\(^1\) established CORDRA, a framework that provided standards for an interoperable repository infrastructure where data repositories were clustered into community federations and their data were retrieved by a global federation using the metadata of each community federation. In 2011, Barcelos \textit{et al.}\(^2\) created an agent-based federated catalog of learning objects (AgCAT system) to facilitate assess of distributed educational resources. Although little machine learning was involved in these two models, their practice of distributed data management and retrieval served as a reference for the development of federated learning algorithms.

In 2012, Balcan \textit{et al.}\(^3\) implemented probably approximately correct (PAC) learning in a federated manner and reported the upper and lower bounds on the amount of communication required to obtain desirable learning outcomes. In 2013, Richtárik \textit{et al.}\(^4\) proposed a distributed coordinate descent method named HYbriD for solving loss minimization problems with big data. Their work provided the bounds of communication rounds needed for convergence and presented experimental results with the LASSO algorithm on 3TB data. In 2014, Fercoq \textit{et al.}\(^5\) designed an efficient distributed randomized coordinate descent method for minimizing regularized non-strongly convex loss functions and demonstrated that their method was extendable to a LASSO optimization problem with 50 billion variables. In 2015, Konecny \textit{et al.}\(^6\) introduced a federated optimization algorithm suitable for training massively distributed, non-identically independently distributed (non-IID) and unbalanced datasets.

In 2016, McMahan \textit{et al.}\(^7\) developed the FederatedAveraging (FedAvg) algorithm that fitted a global model with the training data left locally on distributed devices (known as clients). Experimental results showed that FedAvg performed satisfactorily on both IID and non-IID data and was robust to various datasets. Later, Konecny \textit{et al.}\(^8\) modified the global model update of FedAvg in two ways, namely structured updates and sketched updates, to reduce the uplink communication costs, and experiments indicated that the reduction can be two orders of magnitude. In addition, Bonawitz \textit{et al.}\(^9\) designed the
Secure Aggregation protocol to protect the privacy of each client’s model gradient in federated learning, without sacrificing the communication efficiency. More recently, Smith et al.\textsuperscript{10} devised a systems-aware optimization method named MOCHA that considered simultaneously the issues of high communication cost, stragglers, and fault tolerance in multi-task learning. Zhao et al.\textsuperscript{11} addressed the non-IID data challenges in federated learning and presented a data-sharing strategy whereby the test accuracy could be enhanced significantly with only a small portion of globally shared data among clients. Bagdasaryan et al.\textsuperscript{12} designed a novel model-poisoning technique that used model replacement to backdoor federated learning. Liu et al. used a federated transfer learning strategy to balance global and local learning\textsuperscript{13}.

Most of the previously published federated learning methods focused on optimization of a single issue such as test accuracy, privacy, security or communication efficiency; yet none of them considered the computation load on the clients. This study took into account three issues in federation learning, namely, the local client-side computation complexity, the communication cost, and the test accuracy. We developed an algorithm named Loss-based Adaptive Boosting FederatedAveraging (LoAdaBoost FedAvg), where the local models with a high cross-entropy loss were further optimized before model averaging on the server. To evaluate the predictive performance of our method, we extracted the data of critical care patients’ drug usage and mortality from the Medical Information Mart for Intensive Care (MIMIC-III) database\textsuperscript{14} and partitioned this data into IID and non-IID distributions. In the IID scenario LoAdaBoost FedAvg was compared with FedAvg, while in the non-IID scenario the two methods were both combined with the data sharing strategy and then compared. Our primary contributions include the application of federated learning to health data and the development of the straightforward LoAdaBoost FedAvg algorithm that had better performance than the state-of-the-art FedAvg approach.

**Materials and Methods**

**FedAvg and the data-sharing strategy**

Developed by McMahan et al.\textsuperscript{7}, the FedAvg algorithm trained neural network models via local stochastic gradient descent (SGD) on each client and then averaged the weights of each client model on a server to produce a global model. This local-training-and-global-average process was carried out iteratively as follows. At the $t^{th}$ iteration, a random $C$ fraction of the clients were selected for computation: the server first sent the average weights at the previous iteration (denoted $w_{t-1}^{\text{average}}$) to the selected clients (except for the $1^{st}$ iteration where the clients started its model from the same random weight initialization); each client independently learnt a neural network model initialized with $w_{t-1}^{\text{average}}$ on its local data divided into $B$ minibatches for $E$ epochs, and then reported the learned weights (denoted $w_t^k$ where $k$ was the client index) to the server for averaging (see Figure 1). The global model was updated by the average weights of each iteration.

![Figure 1. Communication between the clients and the server under FedAvg](image)

As demonstrated in the literature\textsuperscript{7}, FedAvg exhibited satisfactory performance with IID data. But when trained on non-IID data, its accuracy could drop substantially in that, with non-IID sampling, stochastic gradient could no longer be regarded as an unbiased estimate of the full gradient according to Zhao et al.\textsuperscript{11}. To solve this statistical problem, they proposed a data-sharing strategy that complements FedAvg via globally sharing a small subset of training data between all the clients (see Figure 2). Stored on the server, the shared data was a dataset distinct from the clients’ data and assigned to clients when FedAvg was initialized. Thus, this strategy improved FedAvg with no harm to privacy and little addition to the communication cost. The strategy had two previously that were $\alpha$, the random fraction of the globally-shared data distributed to each client, and $\beta$, the ratio of the globally-shared data size to the total client data size. Raising these two parameters could lead to a better predictive accuracy but meanwhile make federated learning less decentralized, reflecting a trade-off between non-IID accuracy and centralization. In addition, it is worth mentioning that Zhao et al. also introduced an alternative initialization for their data-sharing strategy: the server could train a warm-up model on the globally shared data and then distribute the model’s
weights to the clients, rather than assigning them with the same random initial weights. In this work, we kept the original initialization method to leave all computation on the clients.

**Figure 2.** FedAvg complemented by the data-sharing strategy: distribute shared data to the clients at initialization

**LoAdaBoost FedAvg**

We devised a variant of FedAvg named LoAdaBoost FedAvg that was based on the median cross-entropy loss to adaptively boost the training process on those clients appearing to be weak learners. The reason for using the median loss rather than average lied in that the latter was less robust to outliers that were significantly underfitted or overfitted client models. Under our approach, not only the model weights but also the cross-entropy losses were communicated between the clients and the server. As demonstrated in Figure 3, at the $t^{\text{th}}$ iteration, the server delivered the average weights $w_{\text{average}}^{t-1}$ and the median loss $L_{\text{median}}^{t-1}$ obtained at the $(t-1)^{\text{th}}$ iteration to each client; then, each client learnt a neural network model in a loss-based adaptive boosting (LoAdaBoost) manner, and reported the learnt weights $w_{k}^{r,t}$ and the cross-entropy loss $L_{k}^{r,t}$ to the server. The global model was parametrized by $w_{k}^{r,t}$.

**Figure 3.** Communication between the clients and the server under LoAdaBoost FedAvg

Our method involved retraining of client models and worked as follows. Same as FedAvg, for Client $k$, its model was initialized with $w_{\text{average}}^{t-1}$ and trained on the local data divided into $B$ minibatches; different from FedAvg, the learning process was performed for $E/2$ instead of $E$ epochs. For odd $E$, $E/2$ would be rounded up to the nearest integer. Here, we used $w_{k}^{r,t}$ and $L_{k}^{r,t}$ to respectively denote the weights and the cross-entropy loss of the trained model and currently $r$ equaled 0. If $L_{k}^{r,0}$ was not greater than $L_{\text{median}}^{t-1}$ (that is, $\triangle L_{k}^{r,0} = L_{k}^{r,0} - L_{\text{median}}^{t-1} \leq 0$), computation on Client $k$ would be finished, with $w_{k}^{r,0}$ and $L_{k}^{r,0}$ sent to the server; otherwise, its model would be retrained for another $E/2$ epochs. Now, the new loss was denoted $L_{k}^{r,1}$ where the superscript 1 indicated the first retraining round. If $\triangle L_{k}^{r,1} = L_{k}^{r,1} - L_{\text{median}}^{t-1} > 0$, the model would be retrained for $E/2 - 1$ more
epochs. This process was repeated for $E/2 - r + 1$ epochs (where $r = 1, 2, 3, \ldots$) and stopped until $\Delta L_k^r = L_k^r - L_{\text{median}}^{r-1} \leq 0$ or the total number of epochs performed on Client $k$ reached $3E/2$. $L_k^0$ and the final $w_k^r$ were sent to the server.

LoAdaBoost was adaptive in the sense that the performance of a poorly-fitted client model after the first $E/2$ epochs was boosted via continuous retraining for a decaying number of epochs. The quality of training was determined by comparing the model’s loss $L_k^r$ with the median loss $L_{\text{median}}^{r-1}$. In this way, our method was able to ensure that the losses of most (if not all) client models would be lower than the median loss at the prior iteration, thereby making the learning process more effective. In addition, because at one iteration only a few of the client models were expected to be trained for the full $3E/2$ epochs, the average number of epochs run on each client would be less than $E$, meaning a smaller local computational load under our method than that of FedAvg. Furthermore, since both $L_{\text{median}}^{r-1}$ and $L_k^r$ were a single value transferred at the same time with $w_k^r$ between the server and Client $k$, little additional communication cost would be incurred by our method.

Similar to other stochastic optimization-based machine learning methods\textsuperscript{11,15–17}, an important assumption for our approach to work satisfactorily was that the stochastic gradient on the clients’ local data was an unbiased estimate of the full gradient on the population data. This held true for IID data but broke for non-IID. In the latter case, an optimized client model with low losses did not necessarily generalize well to the population, implying that reducing the losses through adding more epochs to the clients was less likely to enhance the global model’s performance. This non-IID problem could be alleviated by combining LoAdaBoost FedAvg with the data-sharing strategy, because the local data became less non-IID when integrated with even a small portion of IID data.

In this work, FedAvg was used as the baseline model for IID data evaluation to be compared with LoAdaBoost FedAvg, while FedAvg with the data-sharing strategy was used as the benchmark for non-IID data evaluation to be compared with LoAdaBoost FedAvg with the data-sharing strategy.

The MIMIC-III database
The performance evaluation concerned with the MIMIC-III database\textsuperscript{14}, which contains health information for critical care patients at a large tertiary care hospital in the US. Included in MIMIC-III are 26 tables of data ranging from patients’ admissions, to laboratory measurements, diagnostic codes, imaging reports, hospital length of stay and more. We processed three of these tables, namely ADMISSIONS, PATIENTS and PRESCRIPTIONS, to obtain two new tables as follows:

- ADMISSIONS and PATIENTS were inner-joined on SUBJECT_ID to form the PERSONAL_INFORMATION table which recorded AGE_GROUP, GENDER and the survival status (MORTALITY) of all patients.
- Each patient’s usage of DRUGS during the first 48 hours of stay (that is, STARTDATE – ENDDATE = two days) at the hospital was extracted from PRESCRIPTIONS to give the SUBJECT_DRUG_TABLE table.

Further joining these two tables on SUBJECT_ID gave a dataset of 30,760 examples, from which we randomly selected 30,000 examples to form the evaluation dataset where DRUGS were the predictors and MORTALITY was the response variable. The summary of this dataset was provided in Table 1.

| representation            | count          |
|---------------------------|----------------|
| SUBJECT_ID                | integer: IDs ranging from 2 to 99,999 | 30,000 |
| GENDER                    | binary: 0 for female and 1 for male   | 17,284/12,716 |
| AGE_GROUP                 | binary: 0 for ages less than or equal to 65 and 1 for greater | 13,947/16,053 |
| MORTALITY                 | binary: 0 for survive and 1 for death | 20,841/9,159 |
| DRUGS                     | binary: 0 for not prescribed to patients and 1 for prescribed | 2814 dimensions |

Table 1. Summary of the evaluation dataset

The evaluation dataset was shuffled and split into a training set of 20,000 examples, a test set of 8,000 examples, and a holdout set of 2,000 examples for implementing data-sharing strategy. As with the literature\textsuperscript{7}, the training set was partitioned over 100 clients in two ways: IID in which the data was randomly divided into 100 clients, each consisting of 200 examples; and non-IID in which the data was firstly sorted according to AGE_GROUP and GENDER, and then split into equal-sized 100 clients. Using the skewed non-IID data, we would be able to assess the robustness of our model to scenarios when IID data assumption cannot be made, which is more realistic in the healthcare industry. As for the holdout set, we chose the size of the globally shared dataset $\beta = 5\%$ and the random distributed fraction $\alpha = 4\%$. This means that 1000 examples of the holdout set were shared between the clients, each receiving 40 examples, which were only 0.2% of the total non-IID data. The rationale behind this combination of $\alpha$ and $\beta$ was that, having experimented with different values, we found that under this setting FedAvg complemented by the data-sharing strategy with non-IID data would perform similarly to FedAvg with IID data. Both $\alpha$ and $\beta$ could be increased to further enhance the performance, at the expense of decentralization.
Parameter sets and evaluation metrics
The neural network trained on each client consisted of three hidden layers with 20, 10 and 5 units, respectively, using the rectified linear unit (ReLu) activation functions. There were 56,571 parameters in total. Instead of SGD, the stochastic optimizer chosen in this study was Adaptive Moment Estimation (Adam), which requires less memory and is more computationally efficient according to empirical results\(^1\). We used the default parameter set for Adam in the Keras framework: the learning rate \(\eta = 0.001\) and the exponential decay rates for the moment estimates \(\beta_1 = 0.9\) and \(\beta_2 = 0.999\). In addition, while setting the minibatch size \(B\) to 5, we experimented with the number of epochs \(E = 5, 10\) and 15 and the fraction of clients \(C = 0.1, 0.2, 0.5\) and 0.9.

The evaluation metrics were threefold. First, the area under the ROC curve (AUC) was used to assess the predictive performance of a federated learning model. Here, ROC stands for the receiver operating characteristic curve, a plot of the true positive rate (TPR) against the false positive rate (FPR) at various thresholds. In our study, for a given threshold, TPR was the ratio of the number of mortalities predicted by the global model to the total number of mortalities in the test dataset, while FPR was calculated as \(1 - \text{specificity}\) where \(\text{specificity}\) was the ratio of the number of predicted survivals to the total number of survivals. Second, the communication cost was measured by the number of iterations or, equivalently, communication rounds. One complete round started from the server conveying \(w^{t-1}_{\text{average}}\) under FedAvg (\(w^{t-1}_{\text{average}}\) and \(L^{t-1}_{\text{median}}\) under LoAdaBoost FedAvg) to the clients, and ended with the clients sending \(w^t_k\) under FedAvg (\(w^t_k\) and \(L^t_k\) under LoAdaBoost FedAvg) to the server. Third, we used the average number of epochs on the clients per communication round (denoted \(E_{\text{average}}\)) to measure the local computation load. Under FedAvg, \(E_{\text{average}}\) would be constant since each client learns its model for a fixed value of \(E\). But under our adaptive approach, the number of epochs for each client would be varying and collectively contribute to a different value of \(E_{\text{average}}\) from that of FedAvg. In the experiments, our approach was evaluated against the baseline methods in terms of whether it was able to reach the same AUC target for a non-greater number of communication rounds and with a smaller value \(E_{\text{average}}\).

Results
The experiments began with an illustration of FedAvg’s performance gap between IID and non-IID local data with \(C = 0.1, 0.2, 0.5\) and 0.9 and \(E = 5\) (see Figure 4). Same as the work by McMahan et al\(^7\), each curve in the figure was made monotonically increasing via taking the highest test-set AUC achieved over all previous communication rounds. It is apparent that the global model trained on IID data outperformed that fitted by non-IID data for all different \(Cs\). The former exhibited a higher test-set AUC than the latter (0.79+ versus 0.77+), though they converged after a similar number of communication rounds (for instance, 15+ when \(C = 0.1\)). Since every curve for IID data progressed toward an AUC of 0.79 (depicted as the gray dashed line in Figure 4), we decided to use this value as the target AUC, which each method in evaluation was required to reach before comparing the number of communication rounds and the average epochs per client per round.

![Figure 4. Performance gap between FedAvg with IID and non-IID data](image)

Also noticeable from Figure 4 is that, on the same local data distribution, choosing a smaller fraction of the clients for computation in each round tended to yield better performance. For example, FedAvg on IID data with \(C=0.1\) reached a marginally higher maximum AUC for less communication rounds than with \(C = 0.2, 0.5\) and 0.9, which was consistent with the experimental results reported in the literature\(^7\). Therefore, throughout the performance evaluation we set \(C\) to 0.1. Lastly, to
enable a fair comparison, *FedAvg* and our approach used the same random seeds to select the clients.

**Evaluation on IID data**

As the baseline method, *FedAvg* was compared with *LoAdaBoost FedAvg* under the experimental setup $C = 0.1$ and $E = 5, 10$ and 15. The results are shown in Figure 5 where the horizontal dashed line represents the target test-set AUC of 0.79.

![Figure 5. Evaluation on IID data: performance comparison of FedAvg and LoAdaBoost FedAvg with IID data and $C = 0.1$.](image)

Except the curve for *FedAvg* with $E = 15$, all other curves rose across the dashed target line and the number of communication rounds taken were inversely proportional to $E$. For instance, *FedAvg* with $E = 10$ converged faster than with $E = 5$, and *LoAdaBoost FedAvg* with $E = 15$ faster than with $E = 10$. This was reasonable because adding more epochs on the clients sped up the optimization; however, overfitting could be caused to reduce the performance, as in the case of *FedAvg* with $E = 15$.

Given the same $E$, it can be noted that, although slightly lagging behind at the initial rounds, our method always met the target for less communication rounds than *FedAvg*. A summary of the experiment results proving the superiority of our method is given in Table 2. With $E = 5$, our method reached a test-set AUC of 0.79 for 16 communication rounds and 4.7 average epochs, whereas *FedAvg* took 17 rounds and 5 average epochs. With $E = 10$, the two methods had the same number of rounds but the clients under our method ran 3.8 less epochs on average. With $E = 15$, *FedAvg* failed to fulfill the target but our method continued to improve: only 6 rounds were needed to achieve the AUC of 0.79 for 9.9 average epochs. This indicates that the loss-based adaptive boosting mechanism of our method seemed to have an effective regularization effect that prevented overfitting.

| epochs | communication rounds | average epochs | communication rounds | average epochs |
|--------|----------------------|----------------|----------------------|----------------|
| $E=5$  | 17                   | 5              | 16                   | 4.7            |
| $E=10$ | 9                    | 10             | 9                    | 7.2            |
| $E=15$ | $\infty$            | 15             | 6                    | 9.9            |

Table 2. Evaluation on IID data: average epochs and communication rounds taken to achieve a test-set AUC of 0.79

**Evaluation on non-IID data**

When evaluating on non-IID data, we regarded *FedAvg* complemented by the data-sharing strategy as the benchmark. As aforementioned and shown in Figure 4, the original *FedAvg* algorithm performed undesirably. But when each client was shared with only 0.2% of the total non-IID data, the performance could be increased to an AUC of 0.79+, which was comparable to that of *FedAvg* with IID data. This can be viewed from Figure 6 where the curves for *FedAvg* with shared data for all $C$s converged to a level above 0.79. In addition, same as observed with IID data, a smaller $C$ led to a moderately better performance. Therefore, we set $C = 0.1$ for evaluation as before.

The experiment results for *FedAvg* and our approach with shared data are displayed in Figure 7. The trends of the curves were broadly similar to those discussed for IID data evaluation: higher $E$s would result in faster convergence and *LoAdaBoost*
FedAvg had superior performance to that of FedAvg for all different Es. With $E = 5$, the curve for our method was always above that for FedAvg. With $E = 10$ and 15, the curves fell slightly behind during the first several communication rounds, and then caught up to outperform the curves for FedAvg.

Moreover, all curves for our method achieved a test-set AUC of 0.79, but only when $E = 5$ did the curve for FedAvg fulfill this. Table 3 summarizes the information from Figure 7. LoAdaBoost FedAvg with $E = 5$ took the same number of communication rounds (11) but 0.4 less average epochs than FedAvg. With $E = 5$ and 15 where FedAvg underperformed, our method took 8 and 5 rounds, respectively, and 7.0 and 10.7 average epochs, respectively, to reach the target AUC. These experimental results resonated with our conjecture that LoAdaBoost imposed a beneficial regularization effect on the optimization process. This has particular practical value in learning tasks where the communication cost is the primary concern and more epochs have to be added to the clients to accelerate the model-training.

**Discussion**

Distributed health data in large quantity and of high privacy can be harnessed by federated learning where both data and computation are kept on the clients. In this study, we proposed LoAdaBoost FedAvg, an optimized FedAvg algorithm that adaptively boosted the performance of those clients that were weak learners according to their cross-entropy losses. Experiments
Table 3. Evaluation on non-IID data: average epochs and communication rounds taken to achieve a test-set AUC of 0.79

|     | FedAvg with shared data | LoAdaBoost FedAvg with shared data |
|-----|-------------------------|-----------------------------------|
| E=5 | 11                      | 5                                 |
| E=10| ∞                       | 10                                |
| E=15| ∞                       | 5                                 |

with IID data and non-IID data showed that learning under LoAdaBoost FedAvg converged to slightly higher AUCs, required less client-side computation, and consumed less communication rounds between the clients and the server than FedAvg learning. Our approach can also be extended to learning tasks in other fields, such as image classification and speech recognition, wherever the data is distributed.

As a final point, federated learning with IID data does not always outperform that with non-IID data. For instance, for the language modeling task on the Shakespeare dataset\(^7\), learning on the non-IID distribution reached the target test-set AUC nearly six times faster than on IID. In cases like this, the data-sharing strategy becomes unnecessary and, since the data is severely skewed, LoAdaBoost FedAvg may lose its competitive advantage. In the continuation of our study, we will investigate what kind of medical datasets may result in superior modeling performance with non-IID distribution and why this occurs. Furthermore, we will try to improve the LoAdaBoost FedAvg algorithm to make learning on such datasets easier.

References

1. Rehak, D., Dodds, P. & Lannom, L. A model and infrastructure for federated learning content repositories. In Interoperability of Web-Based Educational Systems Workshop, vol. 143 (Citeseer, 2005).
2. Barcelos, C., Gluz, J. & Vicari, R. An agent-based federated learning object search service. Interdiscip. journal e-learning learning objects 7, 37–54 (2011).
3. Balcan, M. F., Blum, A., Fine, S. & Mansour, Y. Distributed learning, communication complexity and privacy. In Conference on Learning Theory, 26–1 (2012).
4. Richtárik, P. & Takáč, M. Distributed coordinate descent method for learning with big data. arXiv preprint arXiv:1310.2059 (2013).
5. Fercoq, O., Qu, Z., Richtárik, P. & Takáč, M. Fast distributed coordinate descent for non-strongly convex losses. In Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on, 1–6 (IEEE, 2014).
6. Konečnỳ, J., McMahan, B. & Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575 (2015).
7. McMahan, H. B., Moore, E., Ramage, D., Hampson, S. et al. Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629 (2016).
8. Konečnỳ, J. et al. Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016).
9. Bonawitz, K. et al. Practical secure aggregation for federated learning on user-held data. arXiv preprint arXiv:1611.04482 (2016).
10. Smith, V., Chiang, C.-K., Sanjabi, M. & Talwalkar, A. S. Federated multi-task learning. In Advances in Neural Information Processing Systems, 4424–4434 (2017).
11. Zhao, Y. et al. Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018).
12. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. & Shmatikov, V. How to backdoor federated learning. arXiv preprint arXiv:1807.00459 (2018).
13. Liu, D., Miller, T., Sayeed, R. & Mandl, K. Fadl: Federated-autonomous deep learning for distributed electronic health record. arXiv preprint arXiv:1811.11400 (2018).
14. Johnson, A. E. et al. Mimic-iii, a freely accessible critical care database. Sci. data 3, 160035 (2016).
15. Bottou, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT’2010, 177–186 (Springer, 2010).
16. Rakhlin, A., Shamir, O., Sridharan, K. et al. Making gradient descent optimal for strongly convex stochastic optimization. In ICML, vol. 12, 1571–1578 (Citeseer, 2012).
17. Ghadimi, S. & Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. on Optim.* 23, 2341–2368 (2013).

18. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

**Author contributions statement**

L.H initiated the idea, designed the algorithm, processed the data and conducted the experiments. Y.Y conducted the experiments and processed the data. Z.F conducted the experiments and help design the algorithm. S.Z instructed computational optimization and realization. H.D is a clinical expert experienced critical care and provided clinical instructions in this project. D.L initiated the idea, designed the algorithms, supervised and coordinated the project. All authors reviewed the manuscript.

**Additional information**