A comprehensive experimental comparison between federated and centralized learning

Garst, Swier; Dekker, Julian; Reinders, Marcel

DOI
10.1093/database/baaf016

Publication date
2025

Document Version
Final published version

Published in
Database : the journal of biological databases and curation

Database, 2025, baaf016
DOI: https://doi.org/10.1093/database/baaf016
Original Article

A comprehensive experimental comparison between federated and centralized learning

Swier Garst*, Julian Dekker, Marcel Reinders

Intelligent Systems, Delft University of Technology, van Mourik Broekmanweg 6, Delft, Zuid-Holland 2628 XE, The Netherlands

*Corresponding author. Intelligent Systems, Delft University of Technology, van Mourik Broekmanweg 6, Delft, Zuid-Holland 2628 XE, The Netherlands.
E-mail: s.j.f.garst@tudelft.nl

Citation details: Garst, S., Dekker, J. and Reinders, M. A comprehensive experimental comparison between federated and centralized learning. *Database* (2025) Vol. 2025: article ID baaf016; DOI: https://doi.org/10.1093/database/baaf016

Abstract

Federated learning is an upcoming machine learning paradigm which allows data from multiple sources to be used for training of classifiers without the data leaving the source it originally resides. This can be highly valuable for use cases such as medical research, where gathering data at a central location can be quite complicated due to privacy and legal concerns of the data. In such cases, federated learning has the potential to vastly speed up the research cycle. Although federated and central learning have been compared from a theoretical perspective, an extensive experimental comparison of performances and learning behavior still lacks. We have performed a comprehensive experimental comparison between federated and centralized learning. We evaluated various classifiers on various datasets exploring influences of different sample distributions as well as different class distributions across the clients. The results show similar performances under a wide variety of settings between the federated and central learning strategies. Federated learning is able to deal with various imbalances in the data distributions. It is sensitive to batch effects between different datasets when they coincide with location, similar to central learning, but this setting might go unobserved more easily. Federated learning seems to be robust to various challenges such as skewed data distributions, high data dimensionality, multiclass problems, and complex models. Taken together, the insights from our comparison gives much promise for applying federated learning as an alternative to sharing data. Code for reproducing the results in this work can be found at: https://github.com/swiergarst/FLComparison

Introduction

Nowadays, a lot of data is available for the use of machine learning applications. However, in some use cases data do not naturally reside at a single location, and centralizing data might be difficult due to regulation or hardware constraints. One example is medical data (1, 2), which are collected at different hospitals or medical institutions, but cannot leave these institutions due to privacy concerns. In order to make use of this type of data, the concept of federated learning (3) was introduced. Instead of gathering the data in a centralized location before training a single model, in a federated learning environment, the model gets send to wherever the data are available: the so-called clients. In these clients, the models undergo some form of training, after which the updated model parameters are sent back to a central point, referred to as the server. The server then aggregates all the local updates in order to create a new global model, which it then sends back to all the clients, and the cycle repeats. With this setup, only model updates are being communicated, and since the original data never have to leave its origins, the entire process becomes less privacy-sensitive.

The concept of federated learning is analogous to other fields trying to learn from distributed data, particularly distributed optimization (4). Distributed optimization operates similar to federated learning. The major difference is that federated learning generally uses a star network, with a centralized server connected to each client, whereas such a central point is usually not present in distributed learning, in which clients are only connected to (some) other clients. For a more in-depth analysis of distributed learning we refer the interested reader to (5).

A seminal algorithm for federated learning was Federated Averaging (fedAVG, (3)), where a weighted average of the model parameters is taken at the central server each communication round. Since the introduction of federated learning in 2017, a lot of research has been done on its efficiency, performance, and privacy-preserving properties (6, 7). Communication ends up being a bottleneck for efficiency in many federated systems. As a result, techniques have been developed in order to reduce communication rounds (8, 9). With regard to performance, many analyses have been made on the performance of fedAVG (3). These studies focus usually on performance under poorer data distributions (10–12), i.e. where data are not independent and identically distributed (IID) at the different clients, as this is the area where fedAVG performance seems to deteriorate (13). As a result, extensions of fedAVG trying to accommodate for different (non-IID) data distributions have been developed, e.g. (12, 14). Privacy preservation has been explored by means of constructing specific attacks on federated systems (15, 16). As a response, extensions on the original federated algorithms that include some form of increased privacy preservation are becoming a vast area of research (17).

Received 1 May 2024; Revised 28 January 2025

Although all the aforementioned studies have developed federated learning into a vast research area, many analyses remain mostly theoretical. Besides, the aforementioned works assume that the use of a federated approach is given. However, whether to use federated learning is often an important design decision. One example is the medical research field. There have been various efforts over the years to pool medical data from multiple institutes in order to create a large data resource (18, 19). However, the efforts required to create such resources cannot be understated. In these cases, one might want to know whether a federated setup will give comparable performance to central models or whether it is beneficial to create a pooled database.

Recently, papers comparing a federated setting with centralized baselines have been emerging, e.g. (20, 21). However, most comparisons only include one or few classifiers and/or distributions of the data. Therefore, we set out to design a comprehensive set of experiments in which centralized and federated models are compared to one another.

We explore the use of different classifiers (six in total) on multiple datasets (five in total), distributed in several ways. Specifically, we start with a toy problem by comparing performances between federated and centralized classifiers on subsets of the Modified National Institute of Standards and Technology dataset (MNIST), as well as fashion MNIST. Next, we move to more realistic scenarios, with data collected from various locations. These datasets were selected such that they have a variety of data types: tabular (RNA abundance), sequential (kinase activation), and 3D imaging (cardiovascular disease classification). As such, we shed some light on different scenarios in which federated learning might perform similar to a centralized model and when to be careful in assuming such a similar performance.

Methods

Algorithm overview

In total, five different classifiers were implemented in a federated setting: a logistic regressor (LR), a support vector machine (SVM), a fully connected neural network (FNN), a convolutional neural network (CNN), and a gradient-boosting decision tree (GBDT) protocol. All classifiers except the GBDT follow the same federated learning scheme, which is given by Algorithm 1. The federated learning algorithm for the GBDT is given by Algorithm 2.

Algorithm 1. The federated learning algorithm of all classifiers except GBDT

1:	Initialize W_0^g randomly
2:	**For all** r rounds **do**
3:	**On server:** Send W_r^g to all clients i
4:	**On clients:** $W_r^i \leftarrow W_r^g$
5:	**On clients:** $\text{acc}_r^i = \text{acc}(W_r^i, X_{\text{test}}^i, Y_{\text{test}}^i)$
6:	**On clients:** $W_{r+1}^i \leftarrow \text{localStep}(W_r^i, X_{\text{train}}^i, Y_{\text{train}}^i)$
7:	**On clients:** $s^i \leftarrow \text{size}(X_{\text{train}}^i)$
8:	**On clients:** send W_{r+1}^i, acc_r^i, s_i back to server
9:	**On server:** retrieve W_{r+1}^i, acc_r^i and s_i for all i
10:	$W^l = \{W_{r+1}^1, W_{r+1}^2, .., W_{r+1}^i\}$
11:	$S = \{s^1, s^2, .., s^i\}$
12:	**On server:** $W_{r+1}^g = \text{globalStep}(W^l, S)$
13:	**On server:** $\text{acc}_r^g = \text{acc}(W_{r+1}^g, X_{\text{test}}, Y_{\text{test}})$
14:	**end for**

Algorithm 2. The inPrivate learning algorithm

1:	init W^g randomly
2:	for all r rounds
3:	**At server :** $C_a = r \bmod N$
4:	**At server :** send W^g to client C_a
5:	**On client** C_a**:** $\tilde{y}_{\text{pred}} = W^g(X_{\text{train}}^{C_a})$
6:	**On client** C_a**:** $L(C_a) = log(1 - \tilde{y}_{\text{pred}}) + y_{\text{train}} * \log(\frac{\tilde{y}_{\text{pred}}}{1-\tilde{y}_{\text{pred}}})$
7:	**On client** C_a**:** create new tree minimizing $L(C_a)$
8:	**On client** C_a**:** add tree to W^g
9:	**On client** C_a**:** $\text{acc}_r = \text{accuracy}(W^g, X_{\text{test}}^{C_a}, Y_{\text{test}}^{C_{\text{active}}})$
10:	**On client** C_a**:** send W^g, acc_r back to server
11:	**At server:** Receive W^g, acc_r from client C_a
12:	**At server:** $\text{acc}_i^g = \text{accuracy}(W^g, X_{\text{test}} Y_{\text{test}})$
13:	**end for**

The architectures for the FNN and CNN can be found in Supplementary Tables 1 and 2, respectively. A performance comparison with a centralized version of the classifiers was the outcome of interest, rather than optimizing the final performance. Therefore, the neural net architectures were kept simple, allowing for faster training. The linear models were implemented using sklearn's SGDClassifier, using default parameters (except for the loss function). Similarly, the GBDT protocol was implemented using sklearn's GradientBoostingClassifier, using default parameters except for n_estimators, which was initialized as one, and incremented each iteration to allow for the construction of subsequent trees.

Algorithm 1 works as follows: first, a model is initialized on the server with random values. This model is then sent to all clients. Upon reception, every client performs (in parallel) a local training step to update the coefficients of the local model based on stochastic gradient descent (SGD). All clients then send back their updated model to the central server. At the server, these models are combined to update the global model. Then the next epoch starts, and the combined coefficients from the previous round are now used as the new values to be sent out to all clients. In Algorithm 1, the intermediate models are evaluated on the test set at the server (holding a concatenation of all test sets from all clients). Although this is not realistic, we observed no difference between calculating the performance locally at each client and then averaging the performances or doing this centrally on the combined test sets. Therefore, for simplicity reasons, we chose to calculate the performances centrally.

The functions *localStep* and *globalStep* in lines 6 and 12 of Algorithm 1, respectively, are either an implementation of the fedAVG Algorithm (3) or the SCAFFOLD algorithm (12).

The *localStep* for fedAVG is done by means of SGD, i.e.

$$W_{r+1}^i = W_r^i - \eta_l * \nabla L(X_{\text{train}}^i, y_{\text{train}}^i),$$

where W_r^i is the model on client i in round r, η_l the local learning rate, $L(X_{\text{train}}^i, y_{\text{train}}^i)$ is a loss function which differs between classifiers, and X_{train}^i and Y_{train}^i are the training data available at client i. Note that the clients also have to split data into training and testing to be able to independently evaluate a learned classifier. The global step of fedAVG consists of taking

the weighted average for all model parameters:

$$W^g_{r+1} = \frac{1}{S} * \sum_{i=0}^{N} s^i * W^i_{r+1},$$

where s^i is the dataset size of the i^{th} client, N the amount of clients, and $S = \sum_{i=1}^{N} s^i$.

Besides fedAVG, we have also used the SCAFFOLD aggregation algorithm (12). For both the local and global step, SCAFFOLD expands on fedAVG by means of a so-called control variate c. Intuitively, these control variates are used to compensate for model drift, where a model update does not move into the direction of the global optimum due to local datasets being distributed in a non-IID fashion. For the local step:

$$W^i_{r+1} = W^g_r - \eta_l \nabla L(X^i_{\text{train}}, y^i_{\text{train}}) + c^g_r - c^i_r,$$

in which c^g_r and c^i_r are the global and local control variates in round r which are all initialized with zero values. Next (but within the same round), c^i gets updated as

$$c^i_{r+1} = c^i_r - c^g_r + \frac{1}{\eta_l}(W^i_r - W^i_{r+1}).$$

These updated control variates are sent back to the server together with the model parameters and accuracy for that round. Then the global update step of SCAFFOLD first updates the coefficients:

$$W^g_{r+1} = W^g_r + \frac{\eta_g}{N} * \sum_{i=0}^{N}(W^i_{r+1} - W^i_r),$$

with η_g being the global learning rate. The global control variate c^g is

$$c^g_{r+1} = c^g_r + \frac{1}{N} * \sum_{i=0}^{N}(c^i_{r+1} - c^i_r).$$

The localStep of both fedAVG and SCAFFOLD allows for batch learning, i.e. splitting up the local training data into multiple shards, allowing multiple consecutive gradient descent steps each local epoch. However, our main objective was comparison with the central case and not optimizing performance. Therefore, since batch learning was not required for convergence, it was not utilized. Exception to this is the CNN classifier for the kinase datasets, which needed some batch learning for convergence: For all CNN experiments, the local data were split up into 10 batches each epoch.

The federated GBDT is implemented different from the other classifiers, as a decision tree does not lend itself very well for parameter averaging. Instead, for GBDT, we followed the "inPrivate Learning" algorithm from (30), as shown in Algorithm 2. In this algorithm, only one client is active per round. This client is referred to as C_a in Algorithm 2. This client uses the decision trees of its predecessors to calculate a loss on its own dataset, which it then uses to boost the decision tree it builds.

Datasets

All classifiers were tested on five different datasets: (i) MNIST (22), images of handwritten digits; (ii) fashion MNIST (23), images of clothes; (iii) a dataset used for the prediction of Acute Myeloid Leukemia (AML) based on measured gene expressions (28); (iv) a dataset consisting of molecular fingerprints used to predict the activation of certain kinases; and (v) a dataset of 3D Magnetic Resonance Imaging (MRI) scans of hearts from either healthy patients or patient with a cardiovascular disease.

Two and four class MNIST

The first dataset is derived from MNIST (22). In order to reduce complexity, two derivatives were made, using only a subset of the original classes: a two-class problem (MNIST2) and a four-class problem (MNIST4). The digits were chosen based on which combination is most difficult to separate according to the original work introducing MNIST. For MNIST2, we included digits 4 and 9, whereas for MNIST4, digits 2 and 8 are being added. A total of 80% of the dataset is used for training, with the remaining 20% for testing. Three different splits were made, resulting in 6 federated datasets: an IID distribution, a distribution with imbalances in sample size (sample imbalance, SI), and one with imbalances in classes (class imbalance, CI). Figure 1d through 1f and Supplementary Figure 9 show these distributions.

Fashion MNIST

The fashion MNIST dataset (23) was also split into 80% training and 20% test data. Two different distributions were made: an IID distribution and a class-imbalanced distribution. Note that this imbalanced distribution is different from the MNIST distributions as it concerns 10 classes (Figure 2a).

AML dataset (A1–A3)

The third dataset was taken from a study done by Warnat-Herresthal et al. (28). In their experiments, they used measured transcriptomes of gene expressions from patients to predict the presence of acute myeloid leukaemia (AML). These transcriptomes were taken from human peripheral blood mononuclear cells from three different sources. In the datasets, A1 and A2 are transcriptomes that have been measured using microarrays (31), whereas dataset A3 is measured using RNA-sequencing (32). All three datasets contain 12 709 different gene expression values per sample, with labels for 25 different illnesses. Samples were labelled according to (28), i.e. all AML samples were given label (1), and all other labels were joined under one label (0) (in the original work named 'cases' and 'controls'). This approach does result in an inherent class imbalance in the combined dataset, i.e. 7500 samples have label 0 and only 4000 samples have label 1. The datasets were split into 80% training samples and 20% test samples.

Kinase dataset

The kinase dataset was taken from (26). This dataset consists of two deMorgan fingerprints, connectivity and feature based, with as labels the activation values for certain kinases. We chose two of these kinases, being KDR and ABL1. Two of the three datasets were quite sparse, so there is no full overlap between molecules with data on KDR and ABL1, practically creating two datasets. The activation values were binarized between active and non-active, with a cutoff value of 6.3 (e.g. a molecule is seen as activating a kinase if it has a pIC_{50} value higher than 6.3), following the original study.

MRI dataset

The last dataset consisted of 3D MRI scans of patients' hearts, taken from the Multi-Vendor and Multi-Disease Cardiac Segmentation Challenge (MNM) (27), where it was generated as part of a challenge on using deep learning models for cardiac MRI scans. Multiple time frames of MRI scans are available for all patients. Following instructions from (27), only the time points of the end-diastolic and the end-systolic phase were used. As various MRI scanners were used (at different hospitals), not all scans had the same resolution/dimensionality. Scans were brought to equal dimensionality by downsampling to the lowest common dimensions.

We only selected samples from healthy patients and patients with hypertrophic cardiomyopathy (HCM), making it a binary classification problem. The original dataset had a train/validate/test split setup to make the problem extra challenging. Since our main objective is not to solve this issue, but rather compare federated and central classifiers, we opted to redistribute the test/validate/train split to be more similar across all clients (Figure 5b). Note that this did not mix the samples across clients.

Experimental setup

Our goal is to compare a federated classifier performance to a central model, i.e. when all data are available at the server. To do this in a fair way, the learning rate was kept equal between the federated and central runs. Also, the amount of communication rounds was kept equal to the amount of epochs in the central case. Only one 'local' epoch was used during all experiments, meaning that all classifiers sent their updated model back to the server after only one full pass of their local data. Four runs with different model initializations were made per experiment (for both the federated and centralized case). All experiments were executed on one laptop, running all clients as well as the server. The federated learning platform vantage6 (33, 34) was used. Vantage6 uses a dockerized solution, which means that every client (and every task that every node runs) runs in its own docker, therefore being unable to alter the solution of the other clients. The experiment on the MRI dataset, however, was not performed with vantage6 due to size constraints of both data and neural network. For this case, we built a simulated federated environment to run on our high-performance cluster.

Results

Linear models on a binary classification problem show a relation between the learning rate and the amount of clients used

The MNIST dataset (images of handwritten digits) was chosen as a first dataset, as it is known to be a relatively easy problem for modern day classifiers (22). In order to simplify, MNIST was converted into a binary classification problem (MNIST2) by selecting only two of the classes. This dataset was distributed evenly (IID) among 10 clients, meaning that each client had a similar amount of samples, with an even class distribution (Figure 1d, Methods). A logistic regression (LR) and SVM were trained, with varying learning rates for both the central and federated model. Results for the LR model can be found in Figure 1a. These results show that increasing the learning rate leads to faster convergence times, both in the federated and centralized case. There is a consistent factor of 10 between the federated and centralized models, i.e. a federated classifier with a learning rate of 0.5 seems to give a comparable result as a central classifier with learning rate 0.05. This was observed for both the LR and SVM models (results on SVM can be found in Supplementary Figure 6). This can be explained by the amount of clients used, which is also 10. A mathematical motivation behind this is given in Supplement A.1, but the intuition is as follows: a single SGD step in a central model is based on the sum of the loss function over all S available data samples. In a federated setting, these S data points are distributed over N clients, meaning that each client gives an update based on only $\frac{S}{N}$ samples (on average). As these updates are only getting averaged, this is analogous to dividing the learning rate by a factor N.

Extension towards more complex models shows different relationships between federated and centralized models

We were interested whether our findings would be different when more complex classifiers were adopted. Hereto, we examined a fully connected neural net (FNN), a convolutional neural net (CNN), and a decision tree (GBDT) classifier (implementation details in methods). Figure 1b (Supplementary Table 4) shows the area under curve (AUC) values for the accuracy curves (i.e. accuracy versus epochs/communication rounds). This metric is indicative of how the convergence of two classifiers compare, especially if the convergence accuracy is equal, which is shown in the top part of the plot. We ran each classifier four times, resulting in the boxplots in Figure 1b. This figure shows that also for the neural network-based models (FNN and CNN), similar performance is being reached when comparing the federated (IID) and central models of the same classifier. The GBDT shows a slight difference between the federated and central versions. This could be due to the GBDT using a different federated update scheme as all other classifiers (Algorithm 2, methods). The CNN shows some difference in AUC of the accuracy curve between the central and federated setting. This highlights a limitation of this metric, as it is caused by a slightly faster start of convergence in the federated case, as can be seen in Supplementary Figure 7. Furthermore, the convergence accuracy is similar on average, although a higher variance is observed in the central case. Curiously, the relation between the learning rate and the amount of clients that was observed in the first set of experiments does not seem to be present for the neural network-based classifiers, i.e. these had the same learning rate between federated and central experiments. We speculate that this difference is due to the non-convexity of the neural networks.

Binary classification experiments show robustness to sample and class distributions

In the next experiment, the robustness to a change in distribution between clients was tested. Hereto, the MNIST2 dataset was split with a sample imbalance across the clients (SI), i.e. the first client only holds a small fraction of the data, the second client a slightly larger fraction, etc. (Figure 1e). We also considered another imbalance, in which the distribution of classes across clients was skewed (CI) (Figure 1f), i.e. some clients have more samples from one class and others have more samples from the other class, and in between. Figure 1b

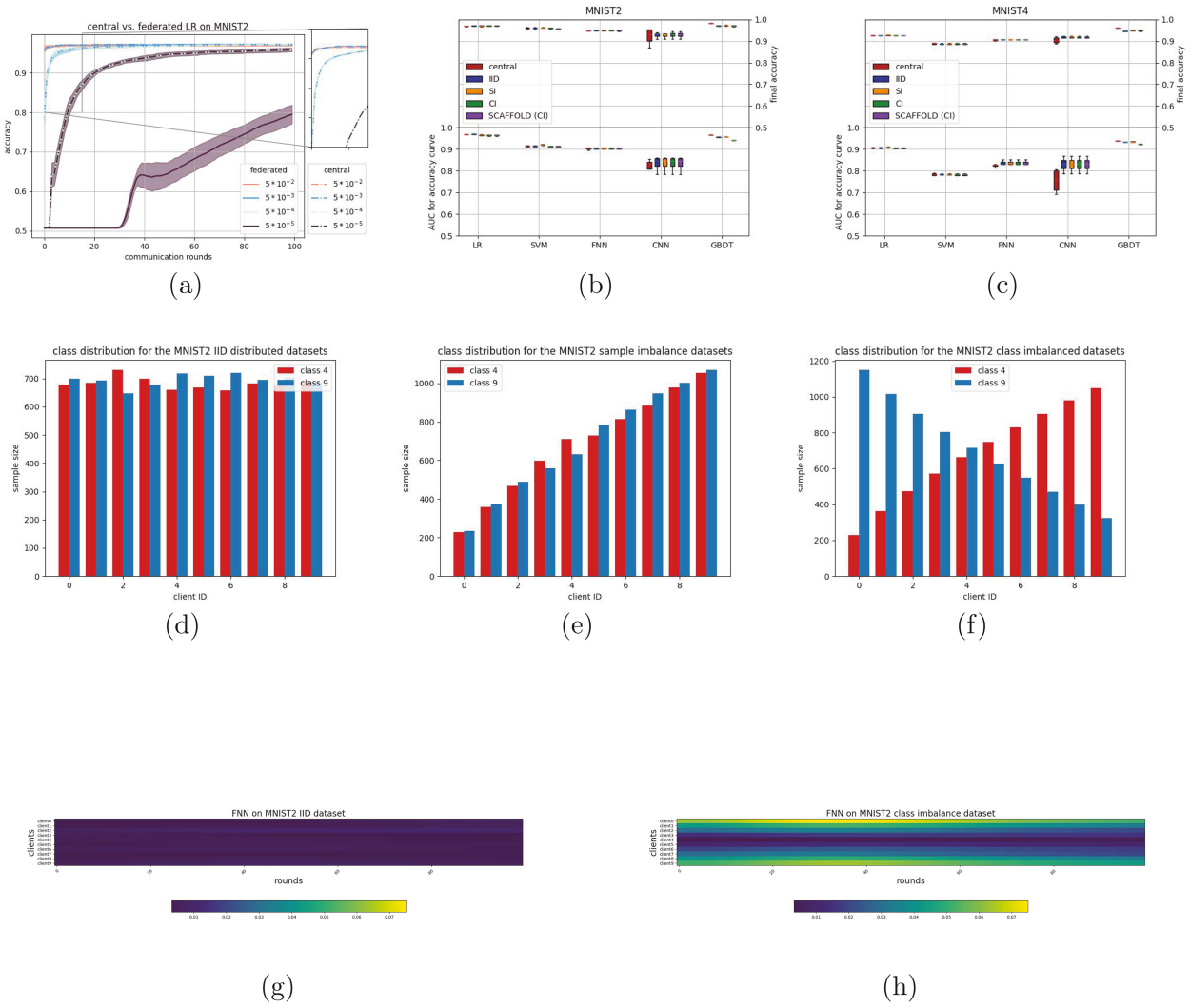

Figure 1. (a–c) show results on MNIST datasets, (d–f) show the various distributions for the MNIST2 dataset (IID, SI, and CI, respectively), and (g–h) show heatmaps resembling model updates throughout training on the IID and CI distributions, respectively.

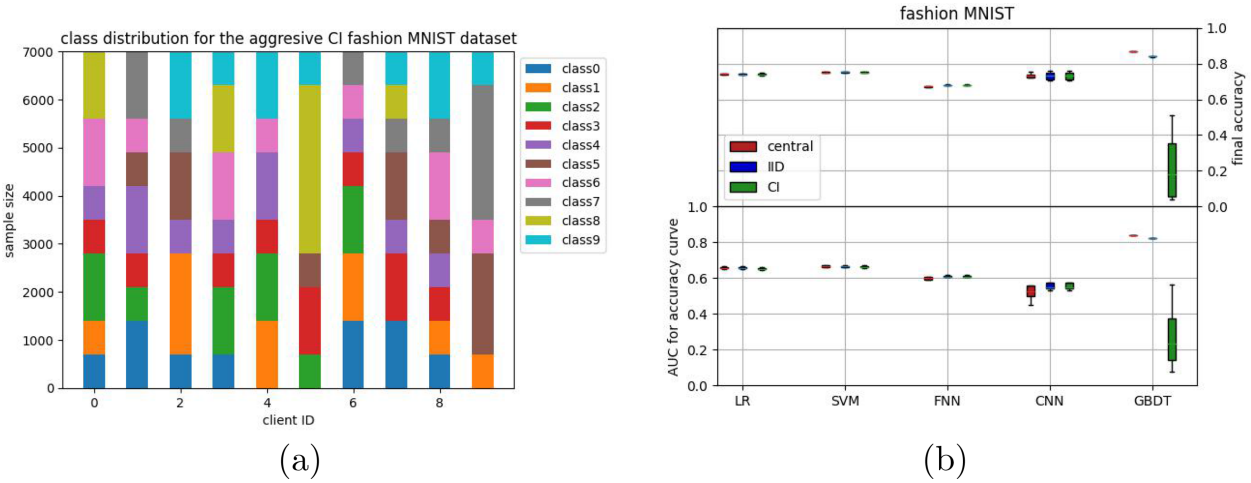

Figure 2. The aggressive CI distribution for the fashion MNIST dataset (a), with the results for both IID and CI distributions reported in (b).

and Supplementary Table 4 show that, with the exception for the GBDT model, similar performances are being attained for the SI and CI settings compared to the IID setting. The SVM does seem to have a slightly higher AUC for the SI distribution, although this is not visible in final accuracy. After observing the accuracy curves (Supplementary Figure 8), this difference seems to come from a slightly faster convergence rate for the SI distribution (about 1 epoch difference).

Although no performance differences are observed in the imbalance settings, we were interested on how both imbalance settings influenced the learning process. Hereto, we visualized the model parameters at the client and at the server at each communication round. In the IID setting (Figure 1g), one can observe that the updated model parameters at the client are more or less similarly changing across the epochs. Interestingly, for the CI setting, this is not the case (Figure 1h). Here, we see that the clients with the larger class imbalance have model updates that are consistently further from the global model.

Extension to multiclass problem results in similar performance as compared to binary classification

To explore the effect of having a multiclass problem, we created a four class dataset out of the original MNIST dataset; MNIST4 (see Methods—datasets). When splitting the samples evenly (IID) across the clients (in number of samples as well as class distributions), the linear models achieve similarly shaped accuracy curves for the federated and central versions, as indicated by the AUC's in Figure 1c (Supplementary Table 4). The neural network-based models show differences in these AUCs. In the case of the CNN, this difference is even visible in the final accuracy. Then we distributed MNIST4 with either sample imbalance (SI) or CI, similar to the two class experiments (see Supplementary Figure 9). The same robustness to the difference in sample and class distributions are observed as in the two class experiments.

SCAFFOLD model updating performs similar to fedAVG model updating

To be more robust to class imbalances across clients, an alternative model updating scheme with respect to fedAVG was introduced: SCAFFOLD ((12), Supplementary Materials A.2). We find that the performance of SCAFFOLD (Figure 1(b) and (c)) is comparable to fedAVG for the generated distributions of the MNIST2 and MNIST4 datasets. This might have been expected, as fedAVG shows similar performance in the CI case compared to the IID case (so no deterioration in performance that could be repaired by SCAFFOLD).

Increasing class distribution aggressiveness gives varying results

After our first experiments on the MNIST datasets, we continued with the fasion MNIST dataset (images of 10 classes of different clothing) as this has been designed to be a harder classification task (23). When distributing evenly across 10 clients (IID), all models behave similarly as for the more simple MNIST classification tasks (Figure 2); there is no visible difference between the central and federated versions of the linear models, the FNN shows a slight difference in AUC but not in final accuracy, and the CNN shows a larger difference in AUC, but not in final accuracy. For the GBDT, some difference can be seen in both AUC and final accuracy.

We then tested a more aggressive class imbalance by distributing the fashion MNIST across clients in such a way that the available classes for each client are not necessarily overlapping, i.e. some clients have training examples of a particular class, whereas other clients do not (Figure 2a). Somewhat surprisingly, differences between the IID setting and the more aggressive CI setting were negligible for most classifiers, except the GBDT, which breaks down for this aggressive CI setting. This might be related to the updating scheme in which the decision tree is updated for each client sequentially (Algorithm 2, methods) instead of simultaneously for all other classifiers (Algorithm 1, methods).

Experiments on high-dimensional data show potential for federated data processing techniques

Until now, all experiments have utilized a single dataset which has then been split into multiple pieces. A more realistic scenario would use multiple datasets, each gathered on a separate client. Hereto, we used datasets that were gathered to predict whether a patient has acute myeloid leukemia (AML) based on their measured transcriptome ((24), Methods). It consists of three different datasets (A1–A3), with 11 500 total patients for which the expression of 12 709 genes are measured. For two of the datasets (A1, A2) the expression is measured using microarray technology, and for the third dataset (A3) this was done using current RNA sequencing technology.

In order to determine the feasibility of learning on these datasets, we first analysed A2 only. A2 was distributed in an IID fashion over 10 clients. As these data have a high dimensionality (over 12.000 features), a dimensionality reduction method seemed to be essential as all classifiers could not perform above random level in the original feature domain (this was also the case for the central models). Therefore, a principal component analysis (PCA) was performed beforehand, reducing the data to 100 features. For this we implemented the PCA in a federated manner (adapted from (25), see Supplementary Algorithm A.3). With the reduced dataset, all classifiers perform similar to what we observed in earlier experiments (Figure 3a). Next, we also tested the SI and CI distributions (before reapplying the federated PCA). Also for this distributed A2 dataset, we observe a high robustness to the different data distributions (Figure 3a and Supplementary Table 5).

Usage of heterogeneous datasets highlights the need to 'think federated'

Then we returned to the combined datasets A1–A3. In this setup, only three clients have been used, each holding one complete dataset. Also in this setting dimensionality reduction using federated PCA was done before training the classifiers. The LR seems to perform fine, although the relation between client amount and learning rate seems to have disappeared (Supplementary Figure 10). However, while the centralized neural network manages to improve slightly over time, the federated neural network now breaks down (Figure 3b). We therefore dived deeper into possible causes of this breakdown.

This breakdown might have been caused by difference in measuring technology, causing the measured features to be

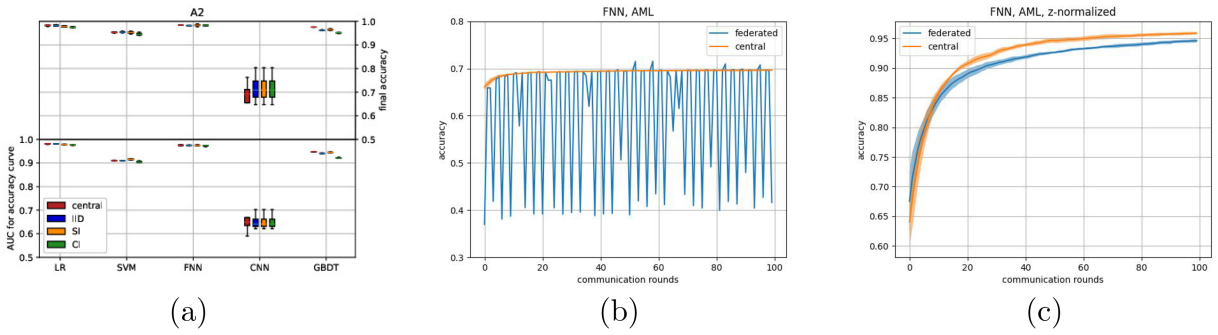

Figure 3. AUC performance of the five classifiers across different splits of dataset A2 across 10 clients in (a), FNN results when all three AML datasets A1–A3 are used with the original data are shown in (b), and (c) shows the FNN results on A1–A3 using the locally scaled data.

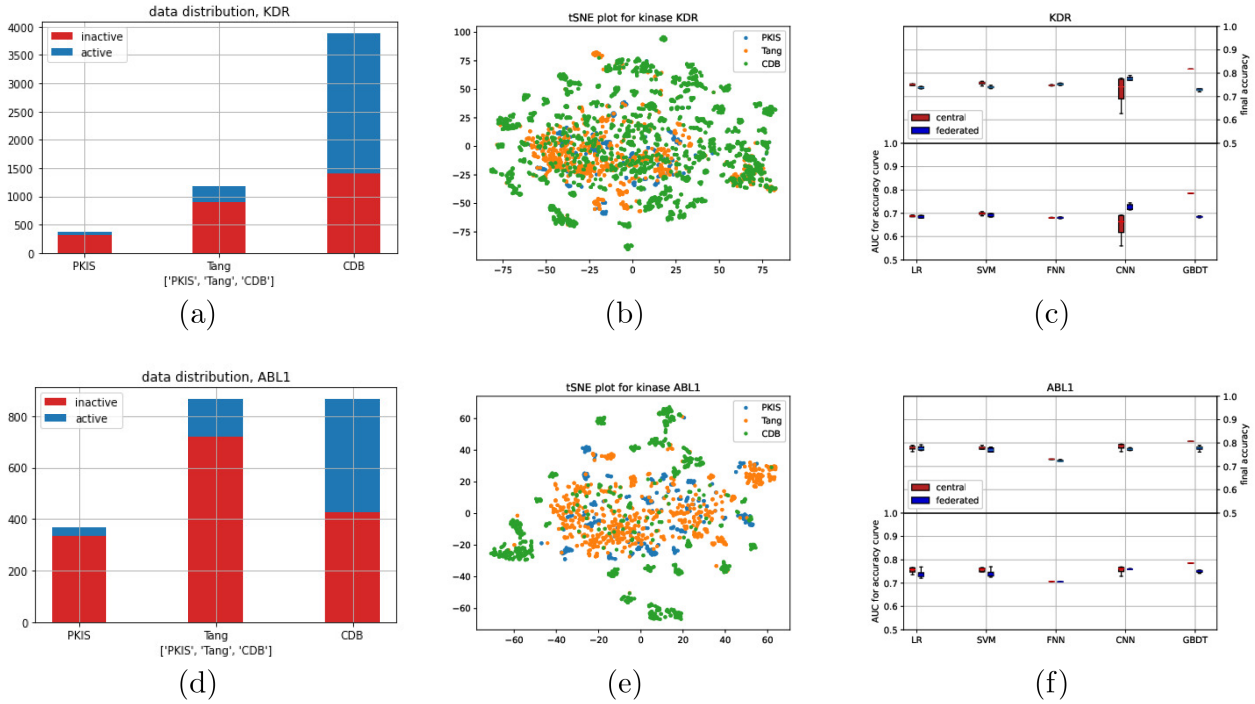

Figure 4. (a) and (d) show data distributions, (b) and (e) show t-distributed stochastic neighbor embedding plots (every dot is a molecule, and color indicates study), and (c) and (f) show performance results for the different classifiers for the central and federated setting.

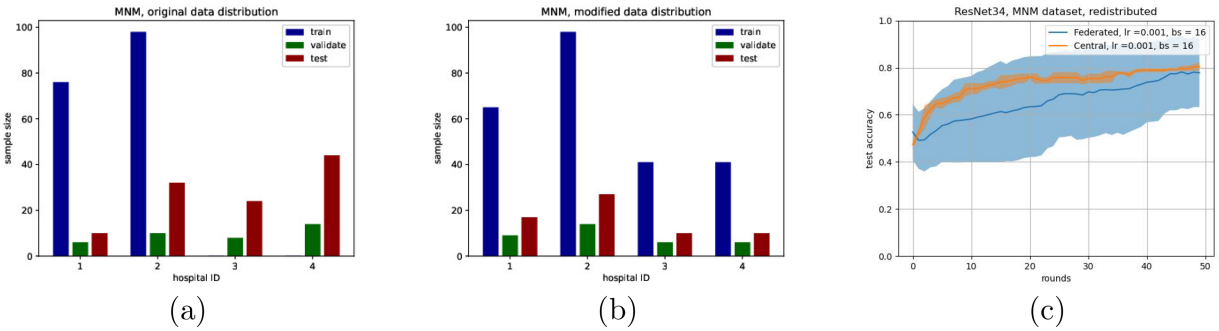

Figure 5. (a) and (b) show distributions of the MNM dataset, originally and after redistribution, respectively, with (c) showing the accuracy curve of ResNet 34 on the distribution from (b) (bs = batch size).

distributed differently between dataset A3 (sequencing technology) and the datasets A1 and A2 (microarray technology). To correct for the different feature distributions between the datasets, we normalized features for all datasets locally, similar to a batch correction scheme, by z-normalizing each gene before applying the federated PCA (Supplementary Algorithm 4). After this batch correction, performance of the federated and central classifiers is similar again (Figure 3c and

Supplementary Figure 9). Although technical batch effect such as these can occur in a centralized setting as well, the nature of federated data being 'unseen' means that such batch effects could go unnoticed. Moreover, it can be tempting to view a federated dataset as one dataset (which happens do be distributed). These results underline the importance of treating distributed datasets as separate entities, instead of as subsets of a larger dataset.

Experiments on 'real-life' distributed datasets show increased variance for federated models

Next, we explored a dataset consisting of molecules that are described by their molecular fingerprints (deMorgan fingerprints, see Methods for details), which can be used to predict the activation or inhibition of certain kinases (26). The dataset is compiled out of three different studies, which we treated as different clients. Two different kinases, KDR and ABL1, were selected as targets for classification, resulting in two datasets. Note that, due to sparsity in the data, these two datasets are not equal in size (for each client), see Figures 4a and 4d for data distributions. After a brief analysis of the datasets, two interesting aspects came to mind. First, the distributions shown in Figure 4a and d could be seen as a combination between a class and sample imbalanced dataset, thereby increasing complexity with respect to earlier distributions. Second, we looked at a combined tSNE plot (Figure 4b and e), which show a large amount of small clusters, most of which are only present at one of the datasets.

Figure 4c and f shows the results for kinase KDR and ABL1, respectively, as well as Supplementary Table 3. For KDR, convergence behaviour seems to be similar, albeit with a higher variance for the linear models, with the exception of CNN and GBDT. However, some difference in final accuracy can be seen for the linear models as well. CNN seems to outperform on KDR with the given metric, which can also somewhat be observed in Supplementary Figure 12. For this experiment, minibatch learning was used for the CNN experiments (both federated and centralized), which could explain this behaviour.

For ABL1, there is a larger difference in convergence for the linear models between the federated and centralized settings. However, final accuracy is still similar, as can also be seen in Supplementary Figure 13. Curiously, both FNN and CNN seem to converge similarly this time, although there is a slightly higher final accuracy for the CNN.

Complex federated dataset shows capability for larger neural networks

To compare performances on a more complex, distributed dataset, we used the publicly available dataset for the MNM challenge from (27). 3D MRI scans of the heart are classified as either a healthy patient or as a patient having one of various cardiovascular diseases. We selected only MRI scans of healthy patients and patients with HCM, which was the largest class of diseases, simplifying the task to a binary classification.

After initial testing, it became clear that the structure of the 3D MRI scans was too complex for the simple linear and shallow neural networks to achieve reasonable performances. We therefore adapted the ResNet-34 model by adjusting the input layer to accommodate the size of the scans, as well as by modifying the output layer to account for the amount of classes. For the federated experiment, the dataset was split based on hospital of origin. This resulted in four clients (the fifth hospital did not have any samples of the classes we selected) (Figure 5a). The train/test split chosen by the original authors proved to be exceptionally challenging without any data augmentation strategies, as performed in earlier work on the same dataset (25, 27). We therefore changed the train/test split as shown in Figure 5b (note that data remained at their original 'location', only local changes to the train/test splits were made). Figure 5c shows the comparison between the federated and centralized experiments. This shows, although the learning behaviour between the two settings is different, both the central and federated approaches converge to a similar accuracy, indicating that federated learning also works well on larger (neural) networks, which is in line with earlier research (25).

Conclusion

This paper describes the results of a set of experiments exploring the differences between centralized and federated machine learning models. Multiple classifiers were used on several datasets, which have been distributed in different ways. Results show that, especially under IID circumstances, a federated classifier could reach similar performances compared to its centralized counterpart. However, a careful choice of parameters such as learning rate and batch size is vital to reach comparable performance. The exact relation between those parameters for reaching equal performance between centralized and federated models remains unclear and could be part of future work, although we have shown that the learning rate can be adapted taking into account the number of clients. Furthermore we have illustrated that when using datasets from multiple sources, one can suffer from batch effects between the datasets. Although this affects both a federated and a central model, we advocate to be extra careful on this aspect in the federated case as the data distributions are not trivially compared in this setting.

For all MNIST variants, fashion MNIST, A2, and both kinase datasets, the same set of classifiers was used. This allowed us to compare performance differences between central and federated classifiers across datasets for all models. It does, however, create the situation that we applied classifiers on data types for which such a specific classifier might not be a good choice. However, since our goal was not to find optimal performance on the studied datasets, but rather compare federated with central models, we deemed this situation acceptable. Nevertheless, our results should not be seen as a recommendation on which type of classifier to use for a specific data type.

The federated GBDT classifier breaks down on the aggressive class-imbalanced distribution of the fashion MNIST dataset, in which each client has approximately half of the total amount of classes. The inPrivate learning algorithm utilizes decision trees created by the previous clients to boost. If these trees were created using a different subset of classes, it can be understood that the resulting gradient does not serve as a helpful tool for the creation of the next decision tree.

For the AML dataset, we have shown the necessity of dimension reduction to achieve reasonable performance for the classifiers studied. We should note that in the original work (28), no form of dimension reduction was used. This might have been possible because they used a deep

neural network including multiple dropout layers, which are known for being helpful at dealing with highly dimensional data (29). In contrast, the neural network used in this paper only consisted of two linear layers, combined with a relu-layer.

One of the strengths of the fedAVG algorithm is that it allows for multiple local epochs, as well as batch learning. This results in multiple learning steps per communication round, potentially reducing the total amount of communication rounds required (3). In our work, we did not explore the amount of local epochs or batch learning in the federated setting. A further analysis on differences between federated and central models could include experiments exploring the influence of varying these parameters.

One limitation of the results on all MNIST datasets is that it is composed of one original dataset. This means that there are no concept shifts (i.e. batch effects) between the different clients, even in the class- and sample imbalanced cases, as can be observed in the AML dataset. Another limitation is the lack of calibration. Calibration is especially interesting in a federated learning case, as there can be a high variance in class distribution between clients (as has been simulated in this work). How this influences a federated classifier is an interesting direction for future work.

Another limitation to this work is the fact that we simulated a federated setting. This was done because it does not affect the performance metrics compared in this work, being the accuracy differences between federated and centralized models. However, for a comparison on runtime, including an analysis of communication overhead for a federated setup, a simulated environment falls short.

Hospitals could decide to 'personalize' their model, i.e. take a model trained using a federated approach (with or without their local dataset), to then retrain that model on their local data only. Although this might lose some generalizability, it could also result in a more accurate model for the institution itself. Exploring such strategies is an interesting venue for future work as well.

Taken together, our work has shown promising results for federated learning as an alternative to centrally organized learning. We strongly advice that, like in central learning with batch effects, it is important to align data distributions across the different clients. Nevertheless, federated learning might open new possibilities to increase data availability for data hungry machine learning methods.

Supplementary data

Supplementary data is available at Database online.

Conflict of interest: None declared.

References

1. Xu J, Glicksberg BS, Su C *et al*. Federated learning for healthcare informatics. *J Healthc Inform Res* 2021;**5**:1–19. https://doi.org/10.1007/s41666-020-00082-4
2. Jochems A, Deist T, van Soest J *et al*. Distributed learning: developing a predictive model based on data from multiple hospitals without data leaving the hospital – a real life proof of concept. *Radiother Oncol* 2016;**121**:459–67.
3. McMahan B, Moore E, Ramage D *et al*. Communication-efficient learning of deep networks from decentralized data. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, USA, Vol. 54, 2017, 20-22 April, pp. 1273–82. https://proceedings.mlr.press/v54/mcmahan17a.html.
4. Nedic A, Ozdaglar A. Distributed subgradient methods for multi-agent optimization. *IEEE Trans Autom Control* 2009;**54**:48–61.
5. Yang T, Yi X, Wu J *et al*. A survey of distributed optimization. *Annu Rev Control* 2019;**47**:278–305.
6. Li Q, Wen Z, Wu Z *et al*. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *CoRR* 2019;**35**:3347–66. http://arxiv.org/abs/1907.09693.
7. Li T, Sahu AK, Talwalkar A *et al*. Federated learning: challenges, methods, and future directions. *IEEE Signal Process Mag* 2020;**37**:50–60.
8. Konecy J, McMahan B, Yu F *et al*. Federated learning: strategies for improving communication efficiency. *CoRR* 2016. http://arxiv.org/abs/1610.05492.
9. Jeong E, Oh S, Kim H *et al*. Communication-efficient on-device machine learning: federated distillation and augmentation under non-IID private data. *CoRR* 2018. http://arxiv.org/abs/1811.11479.
10. Khaled A, Mishchenko K, Richtarik P *et al*. Tighter theory for local SGD on identical and heterogeneous data. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, Vol. 108, pp. 4519–29. PMLR, 2020. https://proceedings.mlr.press/v108/bayoumi20a.html.
11. Haddadpour F, Mahdavi M. On the convergence of local descent methods in federated learning. *CoRR* 2019. http://arxiv.org/abs/1910.14425.
12. Karimireddy SP, Kale S, Morhi M *et al*. SCAFFOLD: Stochastic controlled averaging for federated learning. In: *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119, 2020, pp. 5132–43. https://proceedings.mlr.press/v119/karimireddy20a.html.
13. Zhao Y, Li M, Lai L *et al*. Federated learning with non-IID data. arXiv Preprint abs/1806.00582, 2018.
14. Li T, Sahu AK, Sanjabi M *et al*. On the convergence of federated optimization in heterogeneous networks. In: *Proceedings of Machine Learning and Systems*, Vol. 2, 2020, pp. 429–50. http://arxiv.org/abs/1812.06127201820.
15. Geiping J, Bauermeister H, Dröge H *et al*. Inverting gradients – how easy is it to break privacy in federated learning?. In: *Conference on Neural Information Processing Systems*, Vol. 33, 2020, pp. 16937–47.
16. Shokri R, Stronati M, Shmatikov V *et al*. Membership inference attacks against machine learning models. In: *IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 2016, 22 May, pp. 3–18. http://arxiv.org/abs/1610.05820.
17. Wei K, Li J, Ding M *et al*. Federated learning with differential privacy: algorithms and performance analysis. *IEEE Trans Inf Forensics Secur* 2020;**15**:3454–69.
18. Mahita J, Ha B, Gambiez A *et al*. Coronavirus immunotherapeutic consortium database. *Database* 2023;**2023**:baac112. https://doi.org/10.1093/database/baac112
19. Arora A, Kaur D, Patiyal S *et al*. SalivaDB—a comprehensive database for salivary biomarkers in humans. *Database* 2023;**2023**:baad002. https://doi.org/10.1093/database/baad002
20. Linardos A, Kushibar K, Walsh S *et al*. Federated learning for multicenter imaging diagnostics: a simulation study in cardiovascular disease. *Sci Rep* 2022;**12**:3551.
21. Abdul Salam M, Taha S, Ramadan M *et al*. COVID-19 detection using federated machine learning. *PLoS ONE* 2021;**16**:1–25.
22. LeCun Y, Bottou L, Bengio Y *et al*. Gradient-based learning applied to document recognition. *Proc IEEE* 1998;**86**:2278–324.
23. Xiao H, Rasul K, Vollgraf R *et al*. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR* 2017. http://arxiv.org/abs/1708.07747.
24. Warnat-Herresthal S, Perrakis K, Taschler B *et al*. Scalable prediction of acute myeloid leukemia using high-dimensional machine learning and blood transcriptomics. *Iscience* 2020;**23**:2589–0042.

25. Andreas G. Federated principal component analysis. *Adv Neural Inf Process Syst* 2020;**33**:6453–64.
26. Merget B, Turk S, Eid S *et al*. Profiling prediction of kinase inhibitors: toward the virtual assay. *J Med Chem* 2017;**60**: 474–85.
27. Campello VM, Gkontra P, Izquierdo C *et al*. Multi-centre, multi-vendor and multi-disease cardiac segmentation: the M&Ms challenge. *IEEE Trans Med Imaging* 2021;**40**: 3543–54.
28. Warnat-Herrestal S, Schultze H, Shastry KL *et al*. Swarm learning for decentralized and confidential clinical machine learning. *Nature* 2021;**594**:265–70. https://doi.org/10.1038/s41586-021-03583-3
29. Hinton G, Srivastava N, Krizhevsky A *et al*. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* 2012. http://arxiv.org/abs/1207.0580.
30. Zhao L, Ni L, Hu S *et al*. InPrivate digging: enabling tree-based distributed data mining with differential privacy. In: *IEEE INFOCOM 2018 - IEEE Conference On Computer Communications*, Honolulu, HI, USA, 2018, 15-19 April, pp. 2087–95.
31. Quackenbush J. Microarray analysis and tumor classification. *N Engl J Med* 2006;**354**:2463–72.
32. Mackenzie Ruairi J. RNA-Seq: basics, applications and protocol. https://www.technologynetworks.com/genomics/articles/rna-seq-basics-applications-and-protocol-299461, 2017.
33. Moncada-Torres A, Frank M, Sieswerda M *et al*. VANTAGE6: an open source priVAcy preserviNg federaTed leArninG infrastructurE for Secure Insight eXchange. In: *AMIA Annual Symposium Proceedings Virtual Only*, 2020, 14-18 November, pp. 870–77.
34. Frank M, Sieswerda M, Alradhi H *et al*. Vantage 6 repository, https://doi.org/10.5281/zenodo.3686944 (8 September 2021, date last accessed).

Database, 2025, 00, baaf016, DOI: https://doi.org/10.1093/database/baaf016, Original Article