Machine Learning (2023) 112:633–654
https://doi.org/10.1007/s10994-022-06292-8

Robust federated learning under statistical heterogeneity via hessian-weighted aggregation

Adnan Ahmad[1] · **Wei Luo**[1] · **Antonio Robles-Kelly**[1]

Received: 14 October 2021 / Revised: 1 November 2022 / Accepted: 30 November 2022 /
Published online: 3 January 2023

Abstract
In federated learning, client models are often trained on local training sets that vary in size and distribution. Such statistical heterogeneity in training data leads to performance variations across local models. Even within a model, some parameter estimates can be more reliable than others. Most existing FL approaches (such as FedAvg), however, do not explicitly address such variations in client parameter estimates and treat all local parameters with equal importance in the model aggregation. This disregard of varying evidential credence among client models often leads to slow convergence and a sensitive global model. We address this gap by proposing an aggregation mechanism based upon the Hessian matrix. Further, by making use of the first-order information of the loss function, we can use the Hessian as a scaling matrix in a manner akin to that employed in Quasi-Newton methods. This treatment captures the impact of data quality variations across local models. Experiments show that our method is superior to the baselines of Federated Average (FedAvg), FedProx, Federated Curvature (FedCurv) and Federated Newton Learn (FedNL) for image classification on MNIST, Fashion-MNIST, and CIFAR-10 datasets when the client models are trained using statistically heterogeneous data.

Keywords Federated learning · Model aggregation · Gauss–Newton methods in federated learning

Editors: Krzysztof Dembczynski and Emilie Devijver.

✉ Adnan Ahmad
ahmadad@deakin.edu.au

Wei Luo
wei.luo@deakin.edu.au

Antonio Robles-Kelly
antonio.robles-kelly@deakin.edu.au

[1] School of Information Technology, Deakin University, Geelong, Australia

1 Introduction

Federated learning allows for a centralised model to be trained while the training data remains distributed over a number of clients. Federated learning is important since artificial intelligence applications often demand strict privacy or operate on private data.

One of the early attempts to perform federated learning was FedSGD (federated stochastic gradient descent) (Chen et al., 2016), where the gradients for the model weight updates are averaged proportionally to the number of training samples on each client. FedAvg (federated averaging) (McMahan et al., 2017) builds on FedSGD. In FedAvg, the global model weights are updated, making use of a weighted average. Despite the averaging approaches above are effective for identically distributed data (IID) across local models, real-world applications face several statistical challenges due to non-IID data distributions and unreliable or relatively slow network connections. Moreover, the averaging of the model weights across the clients can lead to accuracy degradation and require a large number of communication rounds in statistically heterogeneous scenarios (Li et al., 2020).

Note that, in non-IID data distributions, classification performance for each data distribution depends on particular parameters that may not share the same indices in all local models. Thus, combining local models with an averaging operation such as that used by FedAvg can cause the model to "drift" from their local objective. Existing FedAvg modifications (Li et al., 2020, 2019; Karimireddy et al., 2020) attempt to reduce client drifts by including additional regularisation terms in local functions, but they do not directly address model variations. FedProx (Li et al., 2020), for example, incorporates a proximal term in the local objective function to avoid sharp deviations from global parameters and aggregates client updates with a standard weighted average approach. We argue, however, that not all parameter estimations are equally important to the model's performance.

In contrast with these methods, we explicitly address this statistical challenge at the aggregation phase of the federated learning process without imposing any penalty in the local objective functions. Here, we propose a novel aggregation approach that considers the quality variations and weighs each parameter of the local client models according to its rate of convergence. This contrasts with the isotropic weighted average used by methods akin to FedAvg. Thus, we compute a weight for each of the client models making use of the Hessian matrix. In this manner, this weight coefficients captures the rate of the convergence of the corresponding model weights. This treatment allows for these parameters to be aggregated according to their convergence rate.

The work present here makes a number of contributions. Firstly, we propose a novel approach for federated model aggregation under non-IID data distributions. At the centre of the approach is an aggregation criterion based on the Hessian metric. The use of the Hessian allows for the aggregation of the client weights to be effected based upon their convergence rate. We demonstrate the superior performance of our proposed approach on MNIST (Lecun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets using statistical heterogeneous distributions better resembling those in real applications. Our aggregation approach is comparable to alternatives such as FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019) on IID data while outperforming the alternatives under non-IID data distributions.

2 Background and motivation

To commence, we require some formalism. Consider a group of K clients working collaboratively to learn a machine learning task. Let $\mathcal{D}_k$ be the training data accessible by client $k \in K$ and n_k be the size of $\mathcal{D}_k$. The aim is to solve the following distributed machine learning problem.

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) \text{where} f(\boldsymbol{w}) = \sum_{k=1}^{K} \frac{n_k}{n} f_k\left(\boldsymbol{w}_k\right) \tag{1}$$

In the equation above, $f(\cdot)$ is the loss function, $\boldsymbol{w}$ are the weights of the global model, $\boldsymbol{w}_k$ are those for the kth client that has access to n_k training samples locally and n denotes the size of total training data distributed across the K clients. Recall that FederatedSGD (FedSGD) (McMahan et al., 2017) was the first federated learning approach where the central server controls the global model, whereby each client performs a single stochastic gradient descent (SGD) step on its training data locally and sends gradient information to the server.

Despite effective, FedSGD is computationally expensive since each client has to communicate twice with the central server after every SGD step. To tackle this drawback, McMahan et al. (2017) propose the Federated Average (FedAvg) algorithm. FedAvg reduces the communication overhead by allowing multiple SGD steps between each communication round. In this manner, instead of just sending a gradient update after each SGD step, FedAvg allows clients to send local parameters estimates at the end of each communication round. As a result, FedAvg (McMahan et al., 2017) is widely used in federated learning, typically involving a central central server that performs aggregation on K locally trained models so as to produce a single global model with weights w.

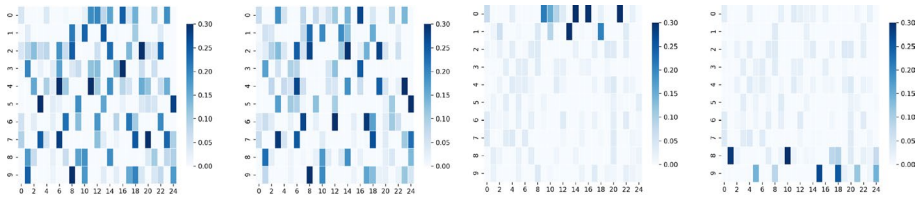
Thus, at each communication round, FedAvg starts by initialising each local model weights with those in the global model and proceeds to train the client's model on its corresponding locally available data $\mathcal{D}_k$. This is done, as usual, by minimising the local loss function $L(\boldsymbol{w}_k)$ making use of stochastic gradient descent(SGD). The central server holding the global model then performs a weighted average across the locally trained models to produce an updated, single global model. This is a simple average over the local model weights. In this manner, the global model weights are then given by

$$\mathrm{w} = \frac{1}{\tau} \sum_{k=1}^{K} n_k \boldsymbol{w}_k, \tag{2}$$

where $\tau = \sum_{k=1}^{K} n_k$ and $\boldsymbol{w}_k$ are the model weights for the kth client.

Therefore, in federated learning methods such as FedAvg and its variants, the contribution of client k is based on the amount of its locally available data as given by n_k. This is important since, for the IID case, all clients have the same amount of data distributed identically across the local clients. This leads to uniformly distributed weights across the clients and, in turn, to an aggregated global model that is evenly balanced with respect to the weights of all local models.

This is an important observation since it underpins the issues arising from model aggregation methods based on simple means over non-IID data distributions. Moreover, in real-world settings, it's not unusual to find data that is not identically distributed over the clients. This setting, where federated learning involves clients with non-IID data distributions, often derives in large model quality variations among local clients. To illustrate this,



(a) IID setting (b) non-IID setting

Fig. 1 Last-layer parameter values in two client models trained on MNIST. In the panels, each column shows the first 25 weights for the 10 output nodes (rows) of the client model. In the non-IID setting, the two client models were trained with only two labels: 0 & 1 (left-hand panel) on the and 8 & 9 on the (right-hand panel)

we show, in Fig. 1 the heatmap for the weights on the last layer of the neural network for two of the local clients used in our experiments when both, IID and non-IID data distributions are used for training. These heatmaps have been computed, making use of the method presented in Eisen et al. (1998) and the type II data distribution presented in Sect. 5. Note that, for both cases, the weights for the clients reflect the data distributions used for training. This is somewhat expected since the performance of each model relies on the parameters trained using the available data to each client. Therefore, data heterogeneity can significantly affect the final value of these trainable parameters.

Since the standard weighted average often used for weight aggregation typically assigns an isotropic weight, $\frac{n_k}{\tau}$ to each local model, it effectively treats each local parameter with equal importance. Thus, the amalgamation of local models with these methods, which employ simple averaging, will prospectively aggregate weights with a larger variance. Moreover, there is a trade-off between communication efficiency and convergence rate in these methods whereby heterogeneity of data slows down the convergence (Li et al., 2020). Consequently, federated learning using simple average aggregation methods may take many communication rounds to converge for heterogeneous data.

3 Hessian-weighted aggregation

The Hessian is a square matrix comprised by the second-order partial derivatives of the loss function. Let the Hessian for the kth client be denoted by $\mathbf{H}_k = \boldsymbol{J}\big(\nabla L\big(\boldsymbol{w_k}\big)\big)$, where $\mathbf{J}\big(\nabla L\big(\boldsymbol{w_k}\big)\big)$ is the Jacobian matrix of the gradient of the loss with respect to the weights w_k of the client under consideration.

The Hessian is fundamental to explain the curvature of the loss function (Pennington & Worah, 2018). Intuitively, the Hessian describes the rate at which a function is being extremized. For example, if the elements of the Hessian at a given value of w_k are large, its an indication that the loss function has a gradient with a high variation along $\boldsymbol{w}_k$. Similarly, if the coefficients of $\mathbf{H}_k$ are small, it implies that the loss function has a critical point at $\boldsymbol{w}_k$. As a result, the Hessian has been widely used in optimisation. Recall that Newton methods employ the Hessian matrix to find the roots of a differentiable function (Nocedal & Wright, 1999). In deep learning, second order training methods leverage the inverse of Hessian to accelerate convergence with, sometimes, significant improvement in performance (Botev et al., 2017). In general, the update step in Newton's method is given by:

$$\boldsymbol{w_{i+1}} = \boldsymbol{w_i} - \mathbf{H}(\boldsymbol{w_i})^{-1}\nabla \boldsymbol{w_i} \tag{3}$$

However, one of the key disadvantages in using the Hessian is its computational cost.

This makes the use of the Hessian often impractical in optimization problems with a large number of parameters (LeCun et al., 2012). As a result, Gauss–Newton optimization method (Becker & LeCun, 1989; Schraudolph, 2001) is often used as a replacement of Newton method to avoid computational cost in highly parameterized optimization problems.

Thus, here we employ an approximation of Hessian making use of the first-order information of the loss function (Chen, 2011). To this end, we follow an often used approach in Gauss–Newton algorithms (Schraudolph, 2001), whereby the Hessian can be approximated by the square of the Jacobian matrix (Nocedal & Wright, 2006; LeCun et al., 2012). Thus, for a given loss function, $L(\boldsymbol{w}_k)$, and making use of the notation above, the approximation of the Hessian is given as follows:

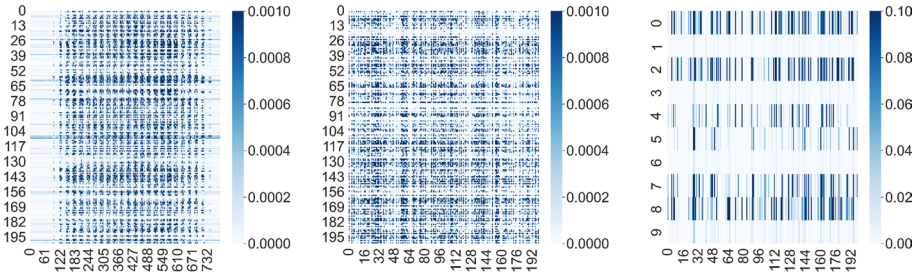
$$\tilde{\mathbf{H}}_k = \mathbf{J}\big(L(\boldsymbol{w_k})\big)^T \mathbf{J}\big(L(\boldsymbol{w_k})\big) \tag{4}$$

Further, following (Zhu et al., 1999), we can use the diagonal of the approximated Hessian above as a scaling matrix (Oren & Luenberger, 1974). This allows us to use the diagonal elements of the approximated Hessian in a manner akin to that employed in Quasi-Newton methods to weight the coefficients for the local models as they are aggregated into the global one. The major advantage of this use of the diagonal terms is that these provide information about the rate of convergence w.r.t weight parameters. The terms with large values indicate that the associated weight parameters are being learned rapidly. On the other hand, the terms that are zero or close to zero imply that their associated weight parameters have already reached critical points. As a result, we can aggregate each local model's weight coefficients according to the diagonal terms of the approximated Hessian $\tilde{\mathbf{H}}_k$.

It is worth noting in passing that we use the Hessian matrix in a very different context as compared to that in which it is being used in second order training methods. In these approaches, which are based Newton's method, the inverse of Hessian is used to improve convergence by finding the steepest descent in the loss landscape. In contrast, here we use the Hessian to moderate the contributions of each client to the global model on particular target classes. This hinges on the notion that the Hessian is related to Fisher Information (Martens, 2020). It is worth noting in passing that the Fisher Information is a Riemannian metric (Amari & Nagaoka, 2000) used as the basis for the natural gradient approach (Amari, 1998; Park et al., 2000) to speed up convergence.

Furthermore the Fisher Information can be interpreted as the expected Hessian of the loss function. Intuitively, if a local model has access to more instances of a specific target, it will be more reliable in its prediction and would be expected to generalise better than that which has access to much less training data of the same class. This will be reflected in the diagonal of Hessian, through its connection to the Fisher Information matrix.

In our proposed approach, we employ the diagonal of the approximated Hessian in the aggregation function. Specifically, we replace the isotropic coefficients, $\frac{n_k}{n}$, in Eq. (2) with the diagonal of $\tilde{\mathbf{H}}_k$. In this setting, each client k approximates the diagonal of $\tilde{\mathbf{H}}_k$ making use of the Jacobian $\mathbf{J}\big(L(\boldsymbol{w}_k)\big)$ and uploads both, the weights $\boldsymbol{w}_k$ and the diagonal terms of the the approximated Hessian to the server holding the global model. If a particular parameter is learned faster on the locally available data, its corresponding term in $\tilde{\mathbf{H}}_k$ will be large



(a) First hidden layer (b) Second hidden layer (c) output layer

Fig. 2 Heat-map obtained from the diagonal of the Hessian of the model parameters for a sample client. In the heat-maps, the Hessian information is computed after 10 epochs of training on the MNIST dataset. All values are normalized before the heat-map computation

and vice versa. In this fashion, the parameters that are learned faster by the local models are treated with higher significance in the aggregation process at the central server.

Algorithm 1 HWA

Server Input : Global model, number of rounds R
for $r \leftarrow 1$ to R **do**
  Communicate Global model weights to all clients $k \in K$
  for each client $k \in K$ in parallel **do**
    Initialize local model with the global one received from the central server
    Train client using SGD
    Compute $\mathbf{D}_k$ using Equation 6
    Communicate the locally trained model and $\mathbf{D}_k$ to the server
  end for
  Aggregate the weights for the hidden layers using Equation 2
  Aggregate the output layer weights using Equation 7
end for
Server Output: Global model

In order to further reduce the communication cost and computational complexity, we only use our approach for the output layer weights $\boldsymbol{w}_o$. This also follows the notion that the feature maps across multiple hidden layers may be sparse, with the output layer often being a dense one due to the fully connected layer right before. Thus, the output layer's Hessian information is more informative than those corresponding to the hidden layers. To illustrate this, in Fig. 2, we show the heat-map obtained from the diagonal of the Hessian model parameters after 10 epochs of training on the MNIST dataset with non-IID Data Type I[1] for the output, second and

[1] For a full description of non-IID Data Type I used in our experiments, please see Sect. 5

first hidden layers of a sample client model. Note that, in contrast with the output layer, the heat-map is very much null for both hidden layers shown. Thus, after receiving the updates from all clients, the central server can perform a simple weighted average on all layers except the output one. Let the output layer weights for the kth client be denoted by $\boldsymbol{w}_k{}^o$, making use of the approximated Hessian, the output layer weights for the global model aggregated out of K clients are given by

$$\mathrm{w}^o = \sum_{k=1}^{K} \frac{\mathbf{D}_k}{\boldsymbol{\beta}_k} \boldsymbol{w}_k^o \tag{5}$$

where

$$\mathbf{D}_k = \mathrm{diag}\Big[\mathbf{J}\big(L\big(\boldsymbol{w}_k^o\big)\big)^T \mathbf{J}\big(L\big(\boldsymbol{w}_k^o\big)\big)\Big] \tag{6}$$

is a diagonal matrix constructed making use of the approximated Hessian and $\beta_k = \| \mathbf{D}_k \|_F + \epsilon$ is a normalisation coefficient. In these expressions, $\| \cdot \|_F$ denotes the Frobenius norm, ϵ is a sufficiently small constant introduced to avoid numerical instability and diag[$\mathbf{Q}$] is an operator that takes at input a square matrix $\mathbf{Q}$ and yields at output a diagonal one whose diagonal corresponds to that of $\mathbf{Q}$.

As mentioned earlier, as the local models converge towards the critical points of the loss function, the diagonal coefficients of the Hessian are expected to become smaller. Therefore, we adopt a hybrid approach, whereby for aggregating particular weight parameters that meet critical points in the local models the average over the client model weights still applies to the global model. Thus, we combine Eqs. (2) and (5) so as to update the output layer weights for the global model as follows:

$$\mathrm{w}^o = \sum_{k=1}^{K} \left(\mathbb{1}\big(\boldsymbol{\beta}_k\big) \frac{\mathbf{D}_k}{\boldsymbol{\beta}_k} \boldsymbol{w}_k^o + \frac{n_k}{\tau} \big(1 - \mathbb{1}\big(\boldsymbol{\beta}_k\big)\big) \boldsymbol{w}_k^o \right), \tag{7}$$

where $\mathbb{1}\big(\boldsymbol{\beta}_k\big)$ is an indicator function $\mathbb{1} : X \rightarrow \{0, 1\}$ defined as:

$$\mathbb{1}(\boldsymbol{\beta}_k) := \begin{cases} 1 & \boldsymbol{\beta}_k \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

In this manner, we can perform robust aggregation while accounting for the possibility that, if the diagonal coefficients of the Hessian become null, i.e. $\boldsymbol{\beta}_k = 0$, the model will still be aggregated as a result of the second term on the right hand side of the expression above.

The algorithm for our method, which we name HWA, for Hessian Weighted Aggregation, is shown in Algorithm 1. Note that, in the algorithm, in order to further reduce the communication cost and computational complexity, we only communicate the diagonal matrix $\mathbf{D}_k$ for the output layer weights $\boldsymbol{w}_k^o$, not the Jacobian $\mathbf{J}\big(L\big(\boldsymbol{w}_k^o\big)\big)$. This can greatly reduce the communication cost since only the diagonal elements of $\mathbf{D}_k$ need to be communicated to the central server.

4 Related work

As mentioned above, work presented in FedCurv (Shoham et al., 2019) does tackle the non-IID case. FedCurv is one of many methods that have been proposed to address statistical heterogeneity. These methods often address the problem by modifying the client's local loss or target function. For instance, FedProx (Li et al., 2020), SCAFFOLD (Karimireddy et al., 2020), FedDANE (Li et al., 2019) and the method of Yao et al. (2019) add regularization terms in the local function to reduce model drifts. FedProx (Li et al., 2020) introduces a proximal term in local function that keeps local parameters close to the global model weights. SCAFFOLD (Karimireddy et al., 2020) applies this proximal term with an additional gradient correction term for strongly convex functions. DANE (Reddi et al., 2016) and its variant (Li et al., 2019) employ gradient correction terms in local functions to address data heterogeneity for both convex and non-convex functions.

Yurochkin et al. (2019) propose a probabilistic federated neural matching (PFNM) algorithm to address permutation invariance properties in multilayer perceptron (MLP) models. Xiao and Cheng (2020) propose a Global Posterior Incorporated Federated Neural Matching (GPI-FNM) algorithm which is another variant of PFNM with an additional KL divergence penalty to integrate global posterior information. Singh and Jaggi (2020) introduce optimal transport (Kantorovich, 2006; Peyré & Cuturi, 2019) in federated learning to match neurons between different models. In a related development aimed at eliminating bias in the global model as induced by the device heterogeneity, Wang et al. (2020) propose Federated normalized averaging (FedNova).

FedNova incorporates a normalized local gradient scheme before sending updates to the server. This treatment prevents global updates from diverging from the global objective. FedLin (Mitra et al., 2021) overcomes device heterogeneity with client-specific learning rates and adds an extra gradient correction term to the local loss function to tackle data heterogeneity.

As mentioned earlier, here we note that local updates, on the other hand, cannot be treated equally since some clients may have more valuable training data than others. Unlike FedNova, our approach captures important information about local data from local updates and prioritizes each update based on its global impact. Moreover, of all the above, FedCurv (Shoham et al., 2019) is the only method that considers the quality variations among local models and employs Fisher Information to evaluate model weights according to their global significance. Specifically, the authors control the regularization term of the local functions with the Fisher Information, which controls each parameter estimate according to its global importance. Here, in contrast, we use the Hessian to adjust the aggregation weights. Moreover, in FedCurv (Shoham et al., 2019), the authors adopt the Elastic Weight Consolidation (EWC) algorithm presented in Kirkpatrick et al. (2017) and incorporate it into the clients' objective functions to control model drift. This is done in conjunction with the standard aggregation approach at the central server to produce a single global model. This is common to all the other methods mentioned above, including FedAvg. This is since the methods which address federated learning over statistically heterogeneous data often do it via a local objective function rather than a modification of the central aggregation scheme.

In our approach, we employ the second-order information provided by the Hessian matrix in the aggregation phase to combine local models into a single global update, directly addressing the impact of the training data's statistical heterogeneity on the information contributed by the clients to the global model. This is also quite different

to FedCurv, where the central server requires access to the Fisher information matrix for all local models with each client involved in the following communication round. Conversely, our method does not reveal the second-order information to the cloud. Thus, there is only one way of communication of the diagonal terms of Hessian in our approach, where each of the clients is agnostic to the gradient information of all the others.

It is worth noting that there have been a number of methods elsewhere in the literature that have employed the Newton method in distributed ML. For example, Islamov et al. (2021) propose both, the Newton-Star and the Newton-Learn methods. These are communication-efficient distributed second order methods for distributed optimization problems. In Islamov et al. (2021), the authors reduce the communication overhead by assuming that the server has at its disposal the second-order information for the clients whereby each of these communicates gradient updates to the server, and the server then takes the Newton step via averaging. Despite being effective, these methods work on distributed optimization problems where data privacy is not a concern and, thus, they do not apply to FL since they require all clients to send their local data to the central server. Building on Islamov et al. (2021), Safaryan et al. (2021) proposed the Federated Newton Learn (FedNL) method. FedNL does not require the communication of training examples to the central server. In FedNL, each client does communicate gradient and compressed Hessian updates to the server, which computes the "global" Hessian by averaging these and then performs the Newton update.

Note, however, that our method is quite different to these in several ways. Firstly, these methods can be regarded as efforts to make Newton methods applicable to distributed optimization problems by employing compression techniques to reduce the communication overhead of the Hessian matrix. Each client hence has to communicate a compressed version of its local Hessian matrix to the server. The central server then employs an average of these local Hessians so as to compute a global Hessian for the Newton updates. Thus, these methods do not explicitly address client drift problems when data is non-IID. Our method, in contrast, does not require the explicit computation of the Newton updates at the central server for the global model parameter update. This is since we aim to address the client drift problem by using the Hessian information for the local models to quantify the contribution of the coefficients of each model for their aggregation to the global one. Secondly, in our method, the server does not aim to compute a global Hessian from the local Hessian matrices, instead it uses each local Hessian independently to govern their contributions to the global model. Thirdly, instead of communicating gradient updates to the server, we employ a first-order optimization method to optimize local models and communicate parameter updates for aggregation. This is a major difference since, by communicating weight updates rather than gradient updates after each local step, local clients can perform local update steps so as to substantially reduce the required communication rounds. This, again, contrast with the methods in Islamov et al. (2021); Safaryan et al. (2021), which require gradient and Hessian updates from all the clients after each local update. Finally, but not least, our method does not explicitly compute the complete Hessian matrix. This is even more important since it is often infeasible to compute and store the complete Hessian of medium-sized and large models. Since our approach is based on the diagonal of Hessian, which we approximate from the first-order information, the memory and computation requirements of our approach allow for large models to be used in practice.

5 Experiments

To evaluate the approach proposed here in cross-silo FL setting, we have followed the experiment setting in McMahan et al. (2017) and employed four publicly available datasets. These are MNIST, Fashion-MNIST, Federated Extended MNIST (FEMNIST) and CIFAR-10. Here, we compare our HWA method with four alternatives. These are FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), FedCurv (Shoham et al., 2019) and the recently proposed FedNL algorithm (Safaryan et al., 2021). Our choice of alternatives stems from the fact that FedAvg (McMahan et al., 2017) is considered as the standard aggregation algorithm for Federated Learning whereas the other two alternatives proposed here explicitly address data heterogeinity. Moreover, recall that FedCurv (Shoham et al., 2019) is an aggregation method which employs the diagonal of the Fisher Information matrix to control the regularization term in the local objective function for each client. FedProx (Li et al., 2020), in the other hand, is a state of the art optimisation approach that introduces a regularization term in the local objective function. Finally, FedNL (Safaryan et al., 2021) is a Newton method that employs the Hessian for the update of the central model.

For all our experiments in cross-silo FL setting, we employ 10 client models with a centralised one and have set the number of communication rounds R to 50 and adopted the assumption that all devices remain active in each round (no client sampling). Here, we consider both the statistically homogeneous (IID) and the statistically heterogeneous (non-IID) settings. In our experiments, these were setup as follows. For IID, data is shuffled and uniformly distributed between all clients. Thus, all clients have an equal amount of training samples from uniformly distributed instances across the 10 classes in MNIST, Fashion-MMINST or CIFAR-10. For non-IID settings, we further consider two different types. For non-IID type I, we use the following procedure to distribute data to each client, which will result in K different partitions. For each class label, we make a random draw from the Dirichlet distribution $Dir(\boldsymbol{\alpha} = \mathbf{0.1})$. The resulting multinomial distribution will determine how many training examples each client will be allocated for that particular class label. Note this is slightly different from the more commonly used per-client random partitioning in Hsu et al. (2019). Note that, our partition procedure is such that allows clients to receive training data which varies not only in terms of class distribution but also in overall number of instances. This contrasts with the approach in Hsu et al. (2019), where the Dirichlet distribution induces variations on a per class basis. For the non-IID type II, we divide the sorted data into 20 equal partitions, where clients are randomly assigned 2 partitions from 2 classes.

Note that FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019) can employ multilayer perceptrons (MLPs) or convolutional neural networks (CNNs) in practice. However, it is practically infeasible to train these highly parameterised models using FedNL (Safaryan et al., 2021) due to the computational resources required. As a result, we have performed two sets of experiments. The first of these employs the standard model architectures in McMahan et al. (2017) to compare our method with FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020) and FedCurv (Shoham et al., 2019). In this setting, for the MNIST and Fashion-MNIST datasets, all the alternatives use a multilayer-perceptron (MLP) which consists of 2-Hidden layers with 200 hidden units, each with ReLu activations. For CIFAR-10, we use a convolutional neural network (CNN) model, which comprises 3-convolutional layers with 3×3 convolutional kernel (channel sizes 32, 64, and 128) followed by 2-fully connected layers.

Table 1 Model accuracy (average ± std) and statistical significance (p-Value) for the MNIST test data set over 10 trials. The p-value shown here corresponds to the one-way analysis of variance (ANOVA)

Epochs	Method	IID	non-IID (I)	non-IID (II)
5	FedAvg	**98.14 ± 0.040**	93.46 ± 1.796	89.46 ± 1.332
	FedProx	98.13 ± 0.026	93.45 ± 1.789	89.27 ± 1.425
	FedCurv	98.10 ± 0.025	93.43 ± 1.770	89.44 ± 1.421
	HWA	98.11 ± 0.066	**94.30 ± 1.255**	**92.49 ± 0.618**
	p-Value	0.676	0.073	0.077×10^{-3}
10	FedAvg	98.09 ± 0.037	90.0 ± 2.415	88.46 ± 0.746
	FedProx	98.08 ± 0.032	93.0 ± 1.658	89.73 ± 0.798
	FedCurv	**98.07 ± 0.025**	90.31 ± 2.384	88.54 ± 0.659
	HWA	98.06 ± 0.020	**93.36 ± 0.747**	**92.18 ± 0.449**
	p-Value	0.93	0.056	0.006×10^{-5}

Bold values indicate the absolute best performance

The second set of experiments shown here involves all the methods under consideration, including FedNL (Safaryan et al., 2021), employing the regularized Logistic Regression (LR) model in Safaryan et al. (2021), with one local update for all methods. Our motivation in this case is to provide a fair comparison to FedNL, where we have used the model proposed by the authors and avoided any advantage the alternatives may have due to their capacity to perform local updates between communication rounds. In this setting we convert the CIFAR-10 dataset into grayscale and resize each image to 28×28 pixels. We used Rank-R ($R = 2$) compression technique for FedNL to communicate Hessian between the central server and the clients.

We first turn our attention to the first of the set of experiments where the client models are either MLPs (MNIST and Fashion-MNIST) or CNNs (CIFAR10). In this case, within each round, the parameters of the global model are copied to each client, and the client is allowed to independently run $E = \{5, 10\}$ epochs to generate local parameter updates and the matrix $\mathbf{D}_k$. Each client then passes these on to the central server to develop the global model as described in Algorithm 1. For all the methods under consideration, the clients were trained with the same batch size of 32 making use of a SGD optimiser with momentum.

We have followed the approach taken by the authors in McMahan et al. (2017), tuning the hyper-parameters for FedAvg via cross-validation and using a learning rate set to 0.001, with 0.001 weight decay and momentum of 0.9. For FedProx, we have followed the experimental setup suggested in Li et al. (2020) whereby we have done a search for the parameter μ over the learning rate values of $\{0.1, 0.01, 0.001\}$. This search yielded a value of 0.01 for the learning rate with a μ of 0.1. Finally, we use the identical settings for FedCurv that we did for our first baseline. As related to FedCurv, note this method has an additional hyper-parameter λ that controls the regularization terms in local loss functions. We have set this by cross validation at at $\lambda = 1$.

Tables 1, 2 and 3 summarise the results obtained in our experiments. In the tables, the best performance is denoted in bold. The tables report the mean and standard deviation, together with the statistical significance (p-Value) of the test accuracy over 10 trails. Note that the methods under consideration are comparable when applied to IID data, with no statistically significant difference between them. However, when applied to non-IID data, HWA outperforms all baselines with a considerable amount. We show the test accuracy and loss as a function of communication rounds in Figures 3, 4, 5 and 6. The learning

Table 2 Model accuracy (average ± std) and statistical significance (p-Value) for the Fashion-MNIST test data set over 10 trials. The p-value shown here corresponds to the one-way analysis of variance (ANOVA)

Epochs	Method	IID	non-IID (I)	non-IID (II)
5	FedAvg	87.38 ± 0.162	72.51 ± 5.368	74.41 ± 3.332
	FedProx	**87.43 ± 0.123**	72.45 ± 5.433	74.12 ± 3.474
	FedCurv	87.39 ± 0.175	72.69 ± 5.115	72.55 ± 4.881
	HWA	87.39 ± 0.176	**76.60 ± 4.278**	**78.25 ± 2.809**
	p-Value	0.944	0.043	0.017
10	FedAvg	88.78 ± 0.132	70.17 ± 5.331	70.0 ± 5.907
	FedProx	88.76 ± 0.139	70.54 ± 8.849	69.62 ± 6.323
	FedCurv	**88.85 ± 0.117**	73.84 ± 5.290	70.04 ± 5.425
	HWA	88.79 ± 0.261	**77.45 ± 3.561**	**76.88 ± 1.410**
	p-Value	0.857	0.049	0.029

Bold values indicate the absolute best performance

Table 3 Model accuracy (average ± std) and statistical significance (p-Value) for the CIFAR-10 test data set over 10 trials

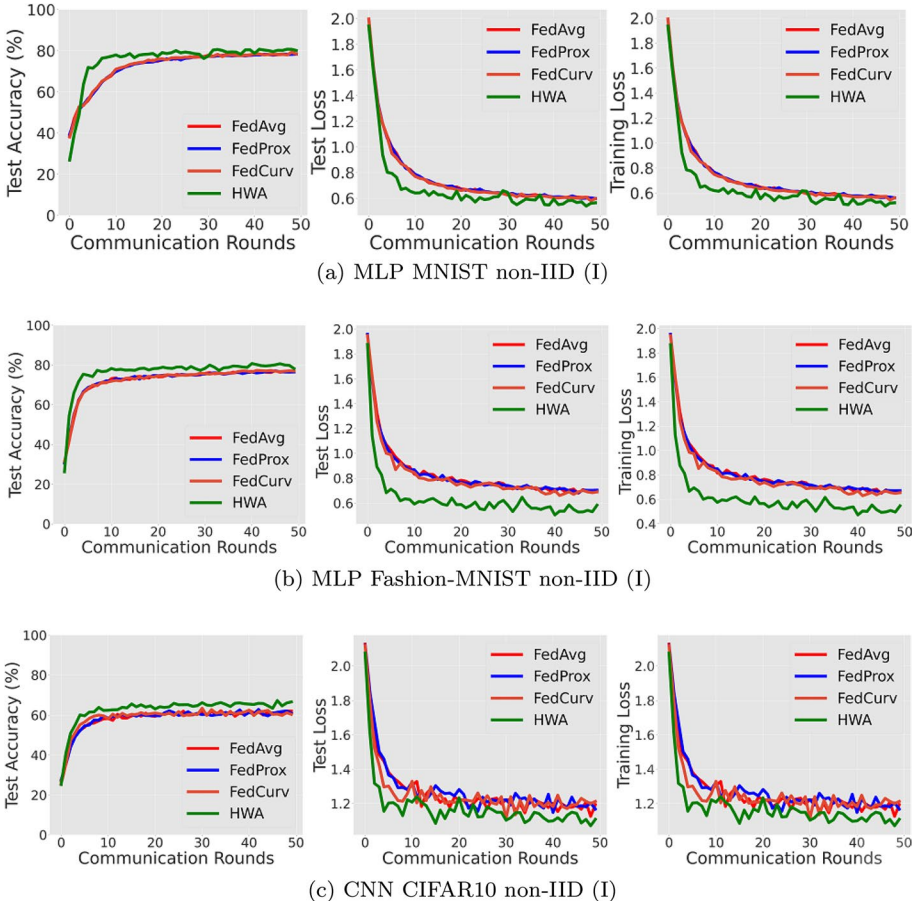
Epochs	Method	IID	non-IID (I)	non-IID (II)
5	FedAvg	77.56 ± 0.416	64.86 ± 1.885	60.86 ± 1.190
	FedProx	**77.61 ± 0.461**	64.53 ± 2.136	59.20 ± 1.055
	FedCurv	77.54 ± 0.497	64.70 ± 2.442	62.03 ± 0.961
	HWA	77.40 ± 0.388	**67.21 ± 1.225**	**63.26 ± 0.947**
	p-Value	0.863	0.03	0.05×10^{-5}
10	FedAvg	**77.99 ± 0.289**	64.05 ± 2.022	58.31 ± 3.116
	FedProx	77.92 ± 0.245	64.74 ± 1.259	59.01 ± 2.328
	FedCurv	77.88 ± 0.237	64.34 ± 2.230	59.62 ± 2.266
	HWA	77.72 ± 0.156	**67.15 ± 1.159**	**62.42 ± 1.504**
	p-Value	0.876	0.092	0.016

Bold values indicate the absolute best performance

curves in the figures show that HWA achieves faster convergence to a better global model. This pattern is consistent across the three datasets and different numbers of local training epochs.

In Figures 3, 4, 5 and 6 we show both, the test accuracy and loss as a function of communication rounds for both, type I and type II non-IID data when 5 and 10 epochs are used for the local training. Note that, when applied to non-IID data, FedAvg and FedProx often perform better and converge faster at 5 rather than 10 epochs within each communication round. This is consistent with the results shown in Tables 1, 2 and 3. This is since, with non-IID data, local models tend to "drift" apart from one another with respect to the global model and adversely affect the aggregation (Li et al., 2020). This behaviour also has an obvious adverse impact on the communication cost, requiring more communication rounds to converge. In comparison, our proposed HWA is much more robust to increasing the number of epochs.

We now turn our attention to the second set of experiments considered here which employs LR as a model for all the alternatives and includes the application of FedNL (Safaryan et al., 2021) to the datasets under consideration. In Figure 7a and b we display the learning curves of LR on the MNIST dataset. Note that, since FedNL is entirely based on the second-order method, it completely outperforms FedAvg and its variants except for



(a) MLP MNIST non-IID (I)

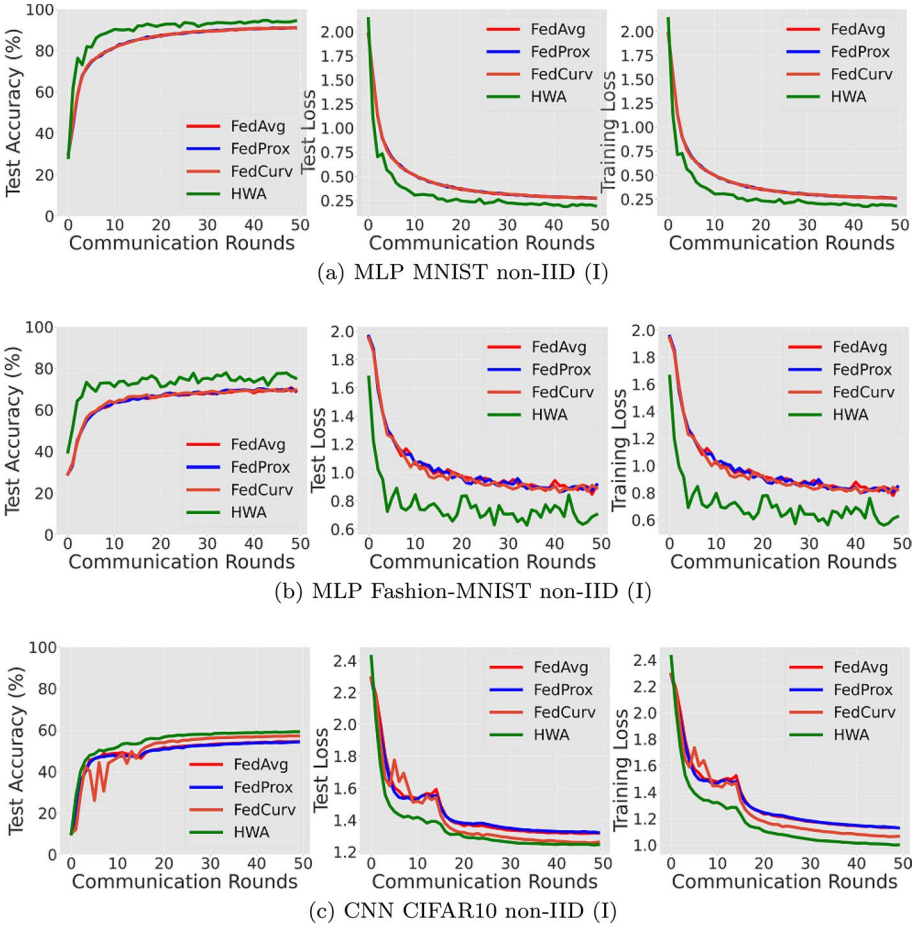
(b) MLP Fashion-MNIST non-IID (I)

(c) CNN CIFAR10 non-IID (I)

Fig. 3 Test accuracy, loss and training loss of the global model as a function of communication rounds when 5 epochs of local training between each communication round are applied on non-IID type I. The two top-most rows show the plots when a multi-layer perceptron (MLP) is trained on the MNIST and Fashion-MNIST datasets, respectively. The bottom row corresponds to the learning curves yielded when a CNN is trained on the CIFAR-10 dataset

HWA. FedNL, however, takes longer to converge than HWA. This behavior is consistent with the accuracy and loss plots for the Fashion-MNIST in Figure 7c and d. In the case of Fashion-MNIST, this is a slightly more complex dataset, whereby the convergence of FedNL deteriorates as compared with the plots for the MNIST dataset. Finally, in Figure 7e and f we show the test accuracy and loss as a function of communication rounds for the alternatives when applied to the resized, monochrome CIFAR10 dataset. Again, as observed in the plots for the other datasets under consideration, FedNL (Safaryan et al., 2021) somewhat struggles to converge. This contrasts with the other methods, of which HWA fairs the best of all.

We now proceed to investigate the degree of heterogeneity in local training data. Note that in previous experiments, we utilised a Dirichlet distribution with $\boldsymbol{\alpha} = \mathbf{0.1}$ to simulate a non-IID distribution. Now we explore the effect of various values of $\boldsymbol{\alpha}$ to see how HWA reacts under extreme data heterogeneous situations. In Figure 8, we illustrate the underlying



(a) MLP MNIST non-IID (I)

(b) MLP Fashion-MNIST non-IID (I)

(c) CNN CIFAR10 non-IID (I)

Fig. 4 Test accuracy, loss and training loss of the global model as a function of communication rounds when 10 epochs of local training between each communication round are applied on non-IID type I. The two top-most rows show the plots when a multi-layer perceptron (MLP) is trained on the MNIST and Fashion-MNIST datasets, respectively. The bottom row corresponds to the learning curves yielded when a CNN is trained on the CIFAR-10 dataset

label distribution using bar plots along with learning curves for all approaches with varied $\boldsymbol{\alpha}$ values. In the bar plot, each bar corresponds to a client and each colour accounts for a class label. At $\boldsymbol{\alpha = 10}$, from the bar plots shown in Figure 8a, we can conclude that the data distribution is close to IID. This is, every client has about the same number of classes and training samples. Therefore, all methods under consideration show equal performance at $\boldsymbol{\alpha = 10}$. Figure 8a–d show that lowering $\boldsymbol{\alpha}$ increases the degree upon which the underlying data distribution diverges from IID becoming more non-IID. Clients are rarely allocated more than three class labels when $\boldsymbol{\alpha = 0.05}$ is used to simulate data heterogeneity. However, even for large degrees of data heterogeneity, HWA not only outperforms the alternatives, but also increases its margin over the other methods under consideration

We now turn our attention to an experimental setting which naturally arises from a cross-device FL environment. To this end, we employ the federated extended MNIST

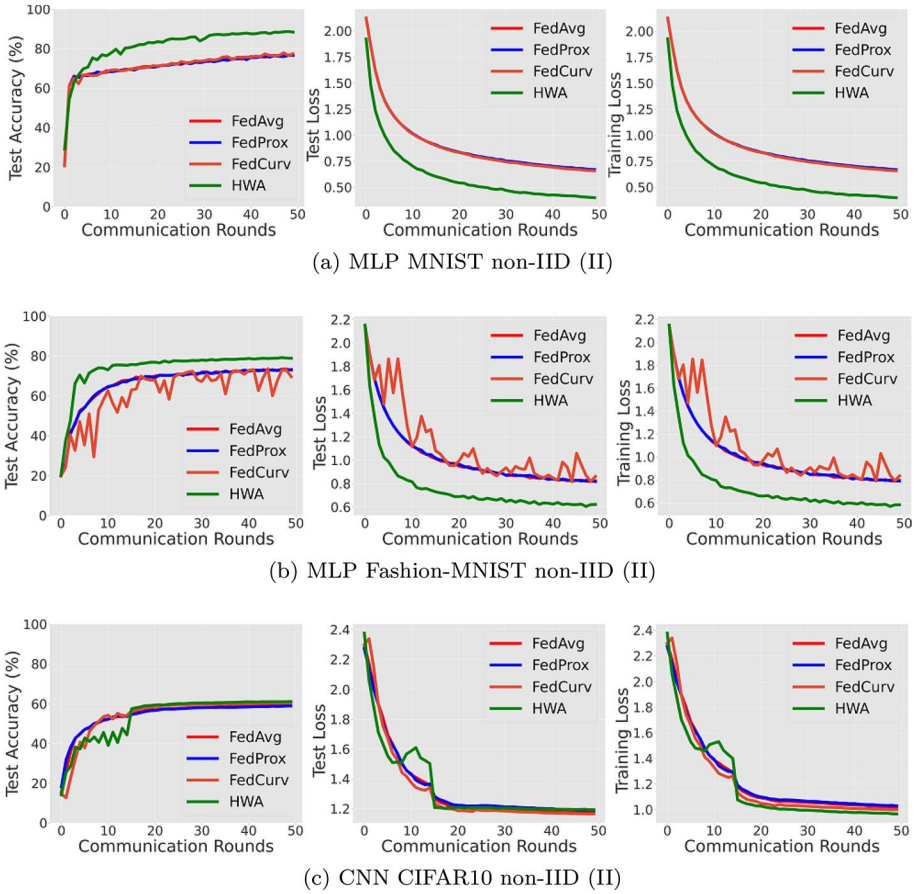
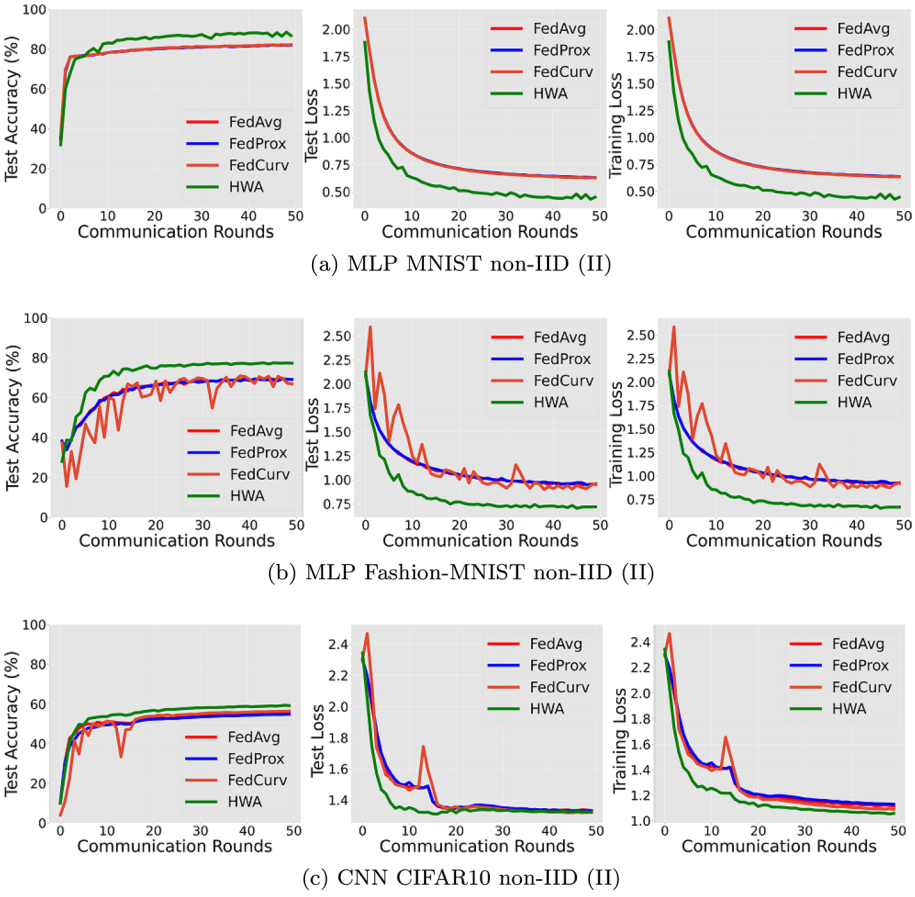


Fig. 5 Test accuracy, loss and training loss of the global model as a function of communication rounds when 5 epochs of local training between each communication round are applied on non-IID type II. The two top-most rows show the plots when a multi-layer perceptron (MLP) is trained on the MNIST and Fashion-MNIST datasets, respectively. The bottom row corresponds to the learning curves yielded when a CNN is trained on the CIFAR-10 dataset

(FEMINIST) dataset obtained from the LEAF benchmark. LEAF is an open-source framework (Caldas et al., 2018) that provides datasets for federated learning. In LEAF, the extended MNIST dataset (Cohen et al., 2017) is partitioned based on the writer of each character/digit. In this way, the partition of the dataset originates from a non-IID distribution whereby each writer has a unique writing style. FEMNIST is comprised of upper and lower-case letters, as well as numerals. As a result, there are a total of 62 classes available. FEMNIST consists of 805, 263 training examples which are distributed among 3, 550 devices.

For the FEMNIST dataset, we have adopted the same model and settings as used in LEAF. The model consists of two convolutional layers with 5×5 convolutional kernel (channel sizes 32 and 64) followed by two fully connected layers. Thus, 5 epochs are used between each communication round and 10 devices are randomly selected for participation in each communication round. Table 4 and Fig. 9 show the performance of



(a) MLP MNIST non-IID (II)

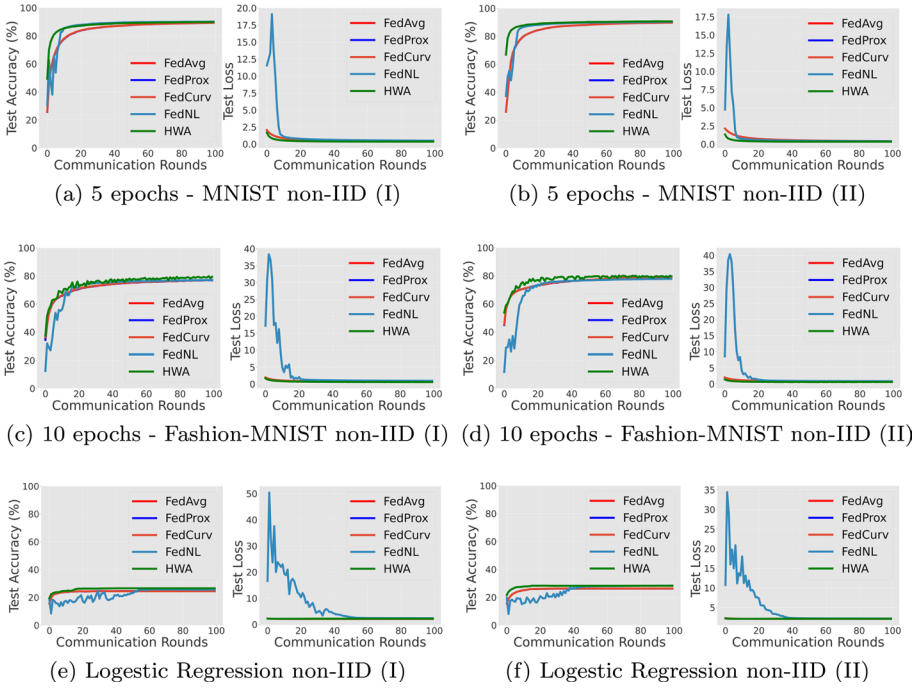
(b) MLP Fashion-MNIST non-IID (II)

(c) CNN CIFAR10 non-IID (II)

Fig. 6 Test accuracy, loss and training loss of the global model as a function of communication rounds when 10 epochs of local training between each communication round are applied on non-IID type II. The two top-most rows show the plots when a multi-layer perceptron (MLP) is trained on the MNIST and Fashion-MNIST datasets, respectively. The bottom row corresponds to the learning curves yielded when a CNN is trained on the CIFAR-10 dataset

methods under consideration. Note that HWA demonstrate superior performance over its alternatives and this trend remain consistent throughout communication rounds. This is consistent with our experiments on the other datasets under consideration.

In order to determine the impact of applying our proposed aggregation scheme to the hidden layers of the model, we further conduct experiments on several variants of our HWA approach and show both the test accuracy and communication cost in Fig. 10. Note the performance of HWA did not improve much when extended to the last hidden layer. On the other hand, the application of the HWA to the whole network tends to degrade the performance. The reason for this could be the sparse nature of feature maps across hidden layers, which yields less informative Hessian information as shown in Fig. 2. Furthermore, note that the communication cost is noticeably greater when the Hessian information for all layers is communicated to the central server. The number of

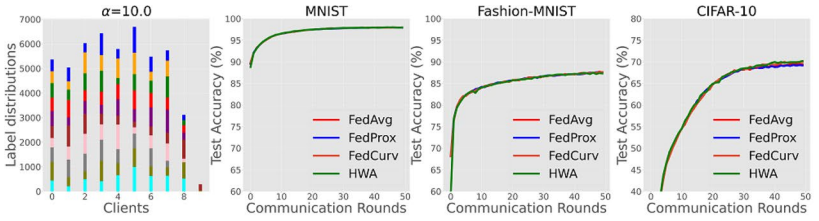


(a) 5 epochs - MNIST non-IID (I)
(b) 5 epochs - MNIST non-IID (II)
(c) 10 epochs - Fashion-MNIST non-IID (I)
(d) 10 epochs - Fashion-MNIST non-IID (II)
(e) Logestic Regression non-IID (I)
(f) Logestic Regression non-IID (II)

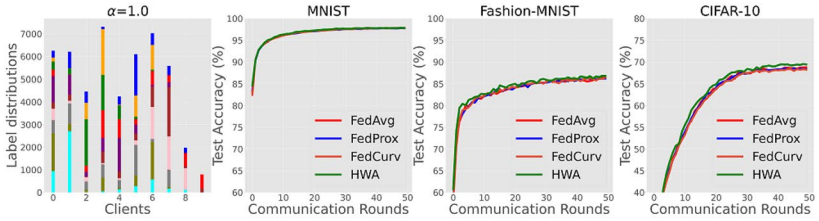
Fig. 7 Test accuracy and loss for the global model as a function of communication rounds. Learning curves yielded when Logistic Regression (LR) is used for the clients and the server

bytes communicated to the central server also increases prominently when the last hidden layer is included with no noticeable increase in performance.

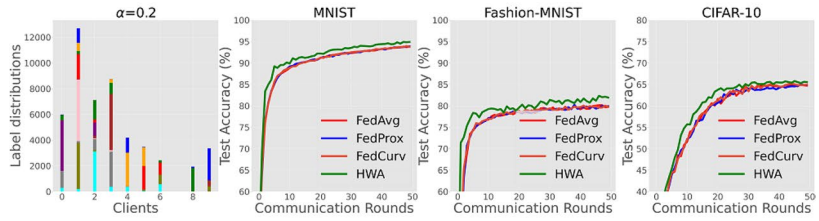
Finally, we investigate both, the robustness of our HWA approach against perturbations in local updates and Standard Differential Privacy at the client level. This is motivated by the notion that each client's concern is to participate in a secure aggregation mechanism and, if necessary, cope with noise perturbations. To this end, we follow Triastcyn and Faltings (2019). To this end, in order to ensure secure communication between FL servers at the client level, each client perturbs its local update with Gaussian noise before communicating to the server. To do this, after each local training step, each client draws a vector equivalent to the size of the model parameters from a Gaussian distribution with zero mean and adds it to its local parameters. In the case of HWA, each client also perturbs its Hessian values as communicated to the server with a Gaussian distribution. In Fig. 11, we show the performance of all the methods under consideration when applied to the CIFAR-10 dataset with a non-IID Data type I whose Dirichlet distribution has been set to $\alpha = 0.01$. The plots show the test accuracy as a function of the noise standard deviation σ. Note that HWA consistently outperforms its alternatives at every value of standard deviation σ, with FedCurv particularly degrading when the standard deviation increases above 0.1.



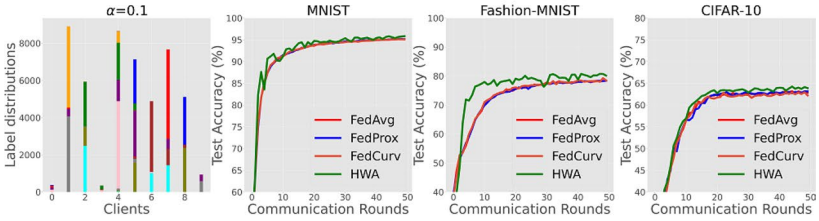
(a) $\alpha = 10$



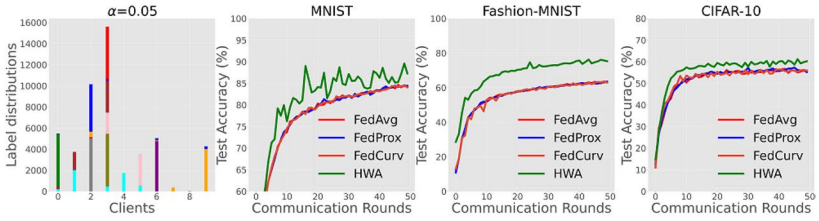
(b) $\alpha = 1$



(c) $\alpha = 0.2$



(d) $\alpha = 0.1$



(e) $\alpha = 0.05$

Fig. 8 Performance of alternatives when global model is trained on non-IID type I with various $\boldsymbol{\alpha}$ values. Left panel shows the underlying distribution of data between clients at $\boldsymbol{\alpha = \{10, 1.0, 0.2, 0.1, 0.05\}}$. Other panels show the test accuracy as a function of communication rounds

Table 4 Model accuracy for the FEMNIST test data set

Epochs	Method	Communication Rounds				
		50	250	500	750	1000
5	FedAvg	29.41	72.10	74.54	74.77	74.87
	FedProx	29.96	71.92	74.46	74.66	74.78
	FedCurv	29.54	72.48	74.82	75.00	75.08
	HWA	**43.85**	**73.16**	**77.66**	**77.83**	**78.02**

Bold values indicate the absolute best performance

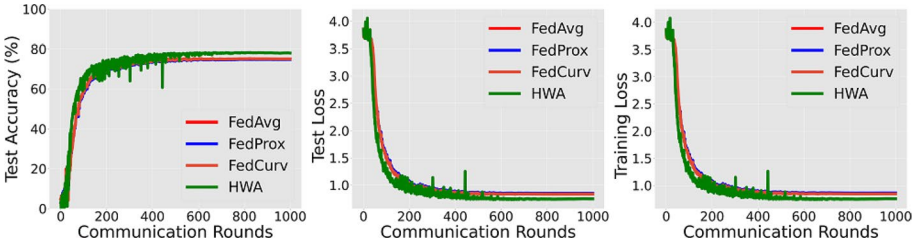


Fig. 9 Test accuracy, test loss and training loss as a function of communication rounds are displayed for the global model. Global model is trained on Federated Extended MNIST (FEMNIST) which is taken from LEAF bench-marking tool

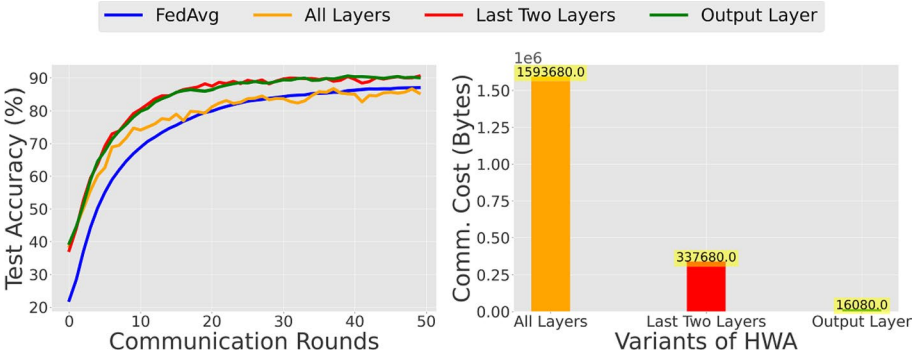


Fig. 10 Illustration of the performance of HWA when the proposed aggregation scheme is applied on different sets of model layers. Left-hand panel: test accuracy for each of these on MNIST dataset with non-IID type I; Right-hand panel: Bar plot showing the number of bytes communicated by a local client to the global server at each round in bytes as a function of the layer-set used

6 Discussion and conclusion

Note that, Among all methods under consideration, FedAvg is the most efficient method in terms of communication and computational complexity. However, FedAvg is prone to client drift problems under statistical heterogeneity. With the same cost of communication bandwidth, FedProx slightly increases the computation complexity since it incorporates a proximal term in the local loss function of each model. Further, FedProx has the

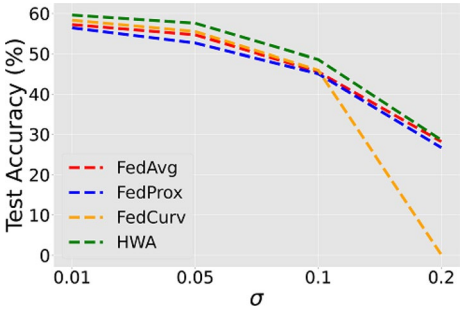


Fig. 11 Accuracy of all the methods under consideration on CIFAR-10 non-IID type I data as a function of increasing Gaussian noise standard deviation σ

drawback of treating all parameters uniformly and penalizing large parameter changes in the central server with an isotropic term. Following the communication and computation complexity of FedCurv as discussed in Shoham et al. (2019), it becomes clear it is the most expensive method. This is since, in FedCurv, the central server and clients require back and forth communication of two additional loss terms equivalent to the size of model parameters. In addition to communication complexity, each client in FedCurv needs to reconstruct the data from the global updates so as to later on use it in the local loss function as a penalty term. This contrasts with HWA, where the clients only need to compute the diagonal of the Hessian of the last layer and communicate it to the server. Note that each client is agnostic about the Hessian information of other clients since only the central server performs the model aggregation. Moreover, a key concern could be the variable communication cost that depends upon the number of classes. Note, however that this would still be considerably less costly than that for alternatives like FedCuv (Shoham et al., 2019) since the output layer often has a small number of parameters as compared to the hidden layers.

Here, we propose a novel approach to address the challenge of statistical heterogeneity among client models in federated learning. To this end, we use the Hessian matrix to weight the aggregation of the client parameters into a global model. This is so as to take into account the varying evidential credence among client models by capturing the impact of data quality variations across local models. We show how this can be effected efficiently using the diagonal of the approximate Hessian. This also leads to a meagre increase in the communication cost, whereby only the diagonal terms of the approximate Hessian have to be shared by the clients with the local server. We also performed experiments that show that our method outperforms Federated Average (FedAvg), FedProx, Federated Curvature (FedCurv) and Federated Newton Learn (FedNL) for image classification on MNIST, Fashion-MNIST, and CIFAR-10 datasets when the client models are trained using statistically heterogeneous data.

Author contributions All authors conceived of the study and the approach. AA implemented the code and carried out the experiments. All authors participated in the design of the algorithm, analysed the experiment results, and helped to draft the manuscript. All authors read and approved the final manuscript.

Funding Not applicable.

Availability of data and material Not applicable.

Code availability Code is available and will be published on https://github.com/ once the paper is accepted.

Declarations

Conflict of interest Not applicable.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation, 10,* 251–276.

Amari, S., Nagaoka, H. (2000). Methods of information geometry.

Becker, S., LeCun, Y. (1989). Improving the convergence of back-propagation learning with second-order methods. Technical Report CRG-TR-88-5.

Botev, A., Ritter, H., Barber, D. (2017). Practical Gauss–Newton optimisation for deep learning. In: *International Conference on Machine Learning*, pp. 557–565.

Caldas, S., Wu, P., Li, T., Konecný, J., McMahan, H. B., Smith, V., Talwalkar, A. S. (2018). Leaf: A benchmark for federated settings. arxiv:1812.01097

Chen, J., Pan, X., Monga, R., Bengio, S., Jozefowicz, R. (2016). Revisiting distributed synchronous SGD. In: *Workshop track - international conference on learning representations*.

Chen, P. (2011). Hessian matrix versus Gauss–Newton hessian matrix. *SIAM Journal on Numerical Analysis, 49*, 1417–1435.

Cohen, G., Afshar, S., Tapson, J., Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In: *2017 International joint conference on neural networks (IJCNN)*, pp. 2921–2926.

Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences, 95*(25), 14863–14868.

Hsu, T. M. H., Qi, H., Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. In: *Neurips workshop on federated learning*.

Islamov, R., Qian, X., Richtárik, P. (2021). Distributed second order methods with fast rates and compressed communication. In: *International conference on machine learning*.

Kantorovich, L. (2006). On the translocation of masses. *Journal of Mathematical Sciences, 133*, 1381–1382.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A. T. (2020). SCAFFOLD: Stochastic controlled averaging for federated learning. In: *International conference on machine learning*, pp. 5132–5143.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*, 3521–3526.

Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

LeCun, Y. A., Bottou, L., Orr, G. B., Müller, K. R. (2012). Efficient backprop. In: *Neural networks: tricks of the trade*.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V. (2020). Federated optimization in heterogeneous networks. In: *Proceedings of machine learning and systems*, vol. 2, pp. 429–450.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A. S., Smith, V. (2019). Feddane: A federated newton-type method. In: *Asilomar conference on signals, systems, and computers*, pp. 1227–1231.

Li, X., xuan Huang, K., Yang, W., Wang, S., Zhang, Z. (2020). On the convergence of FedAvg on non-IID data. In: *International conference on learning representations*.

Martens, J. (2020). New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research, 21,* 1461–1467.

McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.Y. (2017). Communication-efficient learning of deep networks from decentralized data. In: *International conference on artificial intelligence and statistics*.

Mitra, A., Jaafar, R.H., Pappas, G.J., Hassani, H. (2021). Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In: *NeurIPS*.
Nocedal, J., & Wright, S. (2006). *Numerical optimization*. New York: Springer Science & Business Media.
Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Heidelberg: Springer.
Oren, S. S., & Luenberger, D. G. (1974). Self-scaling variable metric (SSVM) algorithms. Part I: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science, 20*(5), 845–862.
Park, H., Amari, S., & Fukumizu, K. (2000). Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks : The Official Journal of the International Neural Network Society, 13*(7), 755–64.
Pennington, J., Worah, P. (2018). The spectrum of the fisher information matrix of a single-hidden-layer neural network. In: *Conference on neural information processing systems*.
Peyré, G., & Cuturi, M. (2019). Computational optimal transport. *Foundations and Trends in Machine Learning, 11,* 355–607.
Reddi, S.J., Konecnỳ, J., Richtárik, P., Póczos, B., Smola, A.J. (2016). Aide: Fast and communication efficient distributed optimization. In: *KAUST research conference 2017: visual computing - modeling and reconstruction*.
Safaryan, M.H., Islamov, R.I., Qian, X., Richtárik, P. (2021). Fednl: Making newton-type methods applicable to federated learning. In: *International conference on machine learning, workshop on federated learning for user privacy and data confidentiality*.
Schraudolph, N.N. (2001). Fast curvature matrix-vector products. In: *ICANN*.
Shoham, N., Avidor, T., Keren, A., Israel, N., Benditkis, D., Mor-Yosef, L., Zeitak, I. (2019). Overcoming forgetting in federated learning on non-iid data. In: *Conference on neural information processing systems, workshop on federated learning for data privacy and confidentiality*.
Singh, S.P., Jaggi, M. (2020). Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33.
Triastcyn, A., Faltings, B. (2019). Federated learning with Bayesian differential privacy. In: *International conference on big data (Big Data)*, pp. 2587–2596.
Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems, 33,* 7611–7623.
Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. In: *CoRR*. arxiv:1708.07747
Xiao, P., Cheng, S. (2020). Probabilistic federated learning of neural networks incorporated with global posterior information. In: *CoRR*. arxiv:2012.03178
Yao, X., Huang, T., Zhang, R., Li, R., Sun, L. (2019). Federated learning with unbiased gradient aggregation and controllable meta updating. In: *CoRR*. arxiv:1910.08234
Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, T.N., Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In: *International conference on machine learning*.
Zhu, M., Nazareth, J. L., & Wolkowicz, H. (1999). The quasi-cauchy relation and diagonal updating. *SIAM Journal on Optimization, 9*(4), 1192–1204.