Enhancing generalization in Federated Learning with heterogeneous data: A comparative literature review

Highlights

Enhancing Generalization in Federated Learning with Heterogeneous Data: A Comparative Literature Review

Alessio Mora,Armir Bujari,Paolo Bellavista

- Present a systematic review of state-of-the-art federated learning approaches proposed to counteract global model performance degradation in presence of heterogeneous data. To this end, we compile an original taxonomy, highlighting the main algorithmic approaches and mechanisms behind each identified category.
- Advancing the current body of knowledge, we empirically evaluate the generalization performance on visual tasks of various methods under moderate and significant levels of data heterogeneity, as common practice in the field. In addition, we benchmark the performance of hybrid techniques, resulting as a combination of client- and server-side algorithmic tweaks, shedding light on some emerging tradeoffs.
- Contribute to the field by providing an open-source code base built in TensorFlow that practitioners and researchers can use to benchmark their solutions against state-of-the-art techniques, readily available at [1].

Enhancing Generalization in Federated Learning with Heterogeneous Data: A Comparative Literature Review

Alessio Mora[a,*], Armir Bujari[a] and Paolo Bellavista[a]

[a]Department of Computer Science and Engineering, University of Bologna, Viale Risorgimento 2, Bologna, 40136, Italy

ARTICLE INFO

Keywords:
Federated Learning
Decentralized Learning
Data Heterogeneity
non-IIDness
Concept drift

ABSTRACT

Federated Learning (FL) is a collaborative training paradigm whereby a global Machine Learning (ML) model is trained using typically private and distributed data sources without disclosing the raw data. The approach paves the way for better privacy guarantees, improved overall system scalability, and sustainability. In this context, Federated Averaging (FedAvg) is a representative FL algorithm adopting a client-server protocol that operates in synchronous rounds, where selected learners contribute to the global model via local model updates, trained using their private data, while a server entity aggregates the local contributions, producing the new-generation global model as a weighted average of the local ones. However, when clients possess (highly) dissimilar data, the FedAvg technique becomes ineffective due to divergence in client models. Consequently, FedAvg-trained models struggle to generalize when presented with unseen data from the global distribution. In this research paper, we conduct a systematic review of state-of-the-art approaches proposed to counteract global model performance degradation in the presence of heterogeneous data.

To this end, we compile an original taxonomy, highlighting the main algorithmic approaches and mechanisms behind each identified category. Advancing the current body of knowledge, we empirically evaluate the generalization performance on visual tasks of various methods under moderate and significant levels of data heterogeneity, as common practice within the surveyed literature. In addition, the paper benchmarks the performance of hybrid techniques, resulting as a combination of client- and server-side algorithmic tweaks, by shedding light on some associated performance tradeoffs. While recognizing other relevant issues in FL, such as device heterogeneity and energy consumption, which have a non-negligible impact on the learning process, these well-investigated topics are not the main focus of this article.

1. Introduction

The vast amount of data being generated at the (far) edge of the Internet has the potential to train highly accurate Machine Learning (ML) models, specifically Deep Learning (DL) models [2], which have brought significant improvements to a variety of intelligent applications and services. However, the sensitivity and confidentiality of collected data pose privacy concerns when managing, storing, and processing data in centralized locations [3].

Federated Learning (FL) promises to alleviate such concerns by enabling a collaborative training process, alternating local computation, and periodic communication. In this setting, data remain on the premises of institutions or organizations that may want to collaborate, as well as, in the case of intelligent applications for user devices (such as smartphones), private preferences or user habits are not disclosed or transmitted elsewhere [4, 5, 6].

Recognizing the benefits of this emergent learning paradigm, many research efforts have been invested on FL, advocating for new model architectures and systems, all sharing a similar behaviour: transmitting ephemeral locally-computed information, such as model parameters or gradients, that are meaningful only with respect to the current state of the learning process (i.e., the current global model parameters). This design contributes to an upgrade of user privacy and meeting legislative requirements, such as the European General Data Protection Regulation (GDPR) [7]. In sensitive domains such as healthcare [8], where data sharing is impeded by regulation (e.g. [9]), FL techniques can enable collaborative learning among institutions without disclosing patients' raw data.

In addition to privacy, FL techniques are also motivated by infrastructural concerns. The vast amount of raw data coming from the network (far) edge to data centers risks overwhelming the network backbone, and consuming parts of this data locally can be more beneficial, as suggested in [10]. Collaborative learning at the (far) edge can also bring as a consequence a reduced carbon footprint compared to cloud-based ML, resulting in a greener technology than data center GPUs, as noted in [11].

Although collaborative learning systems can open up new opportunities for utilizing a wider range of data in data science pipelines, their implementation can be challenging due to the peculiarities of federated environments, including variations in data statistics among learners, the limited and diverse hardware of participating devices, the possibility of still leaking sensitive information during the process, and the complexity of optimizing the process in a distributed environment [4].

When data is uniformly distributed among learning devices, simple methods such as iteratively aggregating clients' model parameters via a weighted average approach (i.e., Federated Averaging [12]) have been demonstrated to produce effective global models, able to reach performance

*Corresponding author
alessio.mora@unibo.it (A. Mora)
ORCID(s): 0000-0001-8161-1070 (A. Mora); 0000-0002-1955-7699 (A. Bujari); 0000-0003-0992-7948 (P. Bellavista)

indicators similar to the centralized counterpart. However, heterogeneous data distributions, which naturally arise in FL environments, hamper the collaborative training process, with clients that tend to overfit local data and, in turn, make aggregation via parameter averaging an ineffective approach.

In this context, we conduct a systematic review of state-of-the-art FL techniques proposed to develop a global model with good generalization ability, despite the presence of dissimilar data among the clients. While an orthogonal line of work focuses on improving local model personalization performance in the presence of non-IID data (e.g., [13, 14, 15]), in this paper, we focus exclusively on methods aimed at enhancing the generalization ability of the global model. We limit the review to methods that extend the canonical FedAvg, i.e. synchronous client-server approaches, in the presence of heterogeneous data, and we do not consider peer-to-peer and/or asynchronous methods (e.g., [16, 17]).

Contributions. The primary original contributions of this paper can be summarized as follows:

1. We accurately discuss the inspiration, rationale, and implementation details of state-of-the-art approaches, proposed to counteract global model degradation.
2. We cluster the surveyed works in coherent groups, by following a novel two-level taxonomy, in a manner that is both technically rigorous and easily understandable.
3. In the second part of the paper, we provide an empirical evaluation of a selection of state-of-the-art algorithms and hybrid approaches, the latter resulting from the combination and joint use of client- and server-side algorithmic tweaks. We empirically evaluate the generalization performance on visual tasks of various methods under moderate and significant levels of data heterogeneity, as common practice in the field.
4. In addition, valuing the spirit of open science and outcome reproducibility, we have created an open-source code base built in TensorFlow that practitioners and researchers can use to benchmark their solutions against state-of-the-art techniques, readily made available at [1].

We are well aware that other significant FL-related surveys exist in the literature: in particular, the work in [18] provides an overview of FL settings, with a focus on the data heterogeneity issue, discussing some approaches aimed at addressing emerging problems. Or the work in [19] classifies data heterogeneity and briefly reviews state-of-the-art methods. Note that also the work in [20] and the work in [21] include an empirical comparison of state-of-the-art algorithms; however, these papers are significantly more limited than this work, considering a more restricted set of algorithms. Furthermore, given the rapid advance in the field, this paper also includes recent literature approaches as well as more accurately present the inspiration and rationale of reviewed methods.

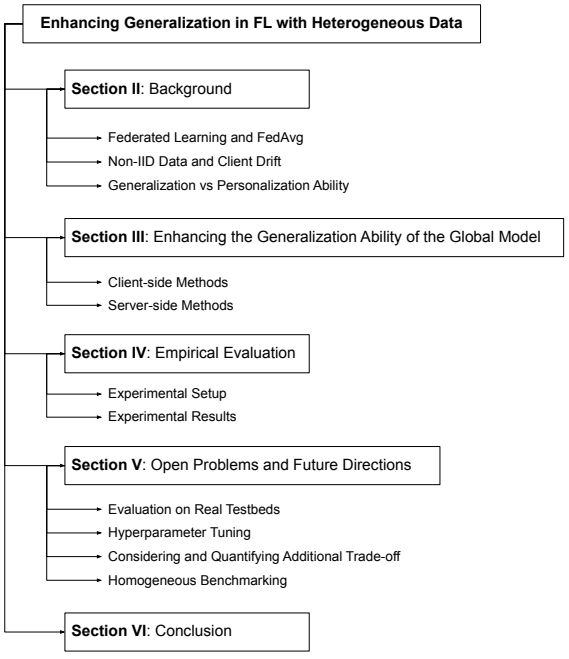


Figure 1: Illustrative organization of the paper.

Organization. In Section 2, we firstly provide a concise overview of the fundamentals of Federated Learning (FL), by delving into key algorithmic aspects. Then, we introduce the *client drift* phenomenon as a consequence of client data heterogeneity, and we detail the difference between *personalization* and *generalization* in FL. In Section 3, we systematically present the most relevant related works, categorizing them primarily as client-side or server-side methods, and further classify them based on their technical inspirations. In Section 4, we propose a benchmark analysis of the most promising approaches, discussing their empirical results. Finally, in Section 5, we outline our vision for future directions and highlight emerging open problems in the field. Finally, in Section 6 the conclusions are drawn. A schematic illustration of the structure of this paper is depicted in Fig. 1.

2. Background

In this section, we present the FL setting and the principal algorithm used as baseline, i.e., Federated Averaging (FedAvg). We also discuss the issue of heterogeneous data distribution among FL clients, which causes a phenomenon known as client drift. Then, we point out the difference between generalization and personalization approaches when designing and assessing the performance of FL methods.

2.1. Federated Learning and FedAvg

At its core, the FL paradigm aims to solve a global objective function expressed as a weighted average over all clients' private datasets. In this setting, there are K clients, with each client k holding a private dataset D_k, the objective

Algorithm 1: FedAvg with clients sending updates Δ instead of complete local models.

Server executes:
  initialize w_0
  for each round $t = 0, 1, 2, 3, ..$
    $c \leftarrow max(C \times K, 1)$
    $S_t \leftarrow$ (random set of c clients)
    for each client $k \in S_t$ **in parallel**
      $\Delta_t^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $\Delta_t \leftarrow \sum_{i \in S_t} \frac{n_k}{n} \Delta_t^k$
    $w_{t+1} \leftarrow w_t + \Delta_t$
ClientUpdate(k, w_t)
  $w \leftarrow w_t$
  $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B)
  for each local epoch e from 1 to E
    for batch $b \in \mathcal{B}$
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    $\Delta \leftarrow w - w_t$
  return Δ to server

is to minimize the function $f(w)$ as defined in Eq. 1,

$$\min_w f(w) := \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \tag{1}$$

where w represents the parameters of the global model to be trained, n_k represents the size of D_k, and n is the total number of examples held by all participating clients. F_k is a local objective function such as cross-entropy loss for a supervised classification task.

FedAvg [12] is a synchronous FL algorithm that adopts a client-server paradigm and is considered the baseline. FedAvg operates in rounds and follows a specific set of steps to solve the optimization problem of Eq. 1, see also Alg. 1. At each round, the global model is sent in broadcast to a random subset of clients, which in turn fine-tune the model locally for a certain number of epochs. Upon process termination, the clients then send back an update to the global model, which is the difference between the locally computed and the received model parameters. The server collects the updates until a time-out, and aggregates the received contributions using weighted averaging based on the number of local data points held by clients. Finally, the server applies the averaged updates to the current global model. In this aggregation step, the inverse of the averaged updates can be seen as a pseudo-gradient, and the server can employ an optimizer of choice to apply such pseudo-gradients to the previous-generation global model. FedAvg employs Stochastic Gradient Descent (SGD) with a learning rate of 1 as server-side optimizer [22].

It is worth noting that in federation scenarios that include hardware-constrained devices, a subset of the activated clients may struggle to send back their updates in time, before the round ends. Such late contributions would be discarded, raising issues of fairness [23]. To address this aspect, asynchronous versions of FedAvg have been proposed in the literature (e.g., [16, 24, 25, 17]), where the server and the clients announce their updates and model weights asynchronously. While recognizing the operational benefits of these approaches in the literature, in this paper we focus our attention on more general and widespread solutions that consider synchronous FedAvg as a baseline approach.

2.2. Non-IID Data and Client Drift

In classic distributed deep learning tasks, such as long-lived tasks running in datacenter environments, the typical assumption is that training data on worker machines are independently and identically distributed (IID), since data are centralized and directly accessible and can be re-partitioned across worker nodes. However, in FL settings, this assumption does not hold as local data on clients may not accurately represent the global distribution (i.e., the combination of local datasets). As a result, non-IIDness of local data leads to inconsistencies between the local objective of each FL participant and the global optima [26]. When fine-tuning the model on private data, the client model deviates from the global model received at the start of the round and from models of other clients that hold data drawn from a different distribution. This deviation is known as client drift and is depicted in Figure 2.

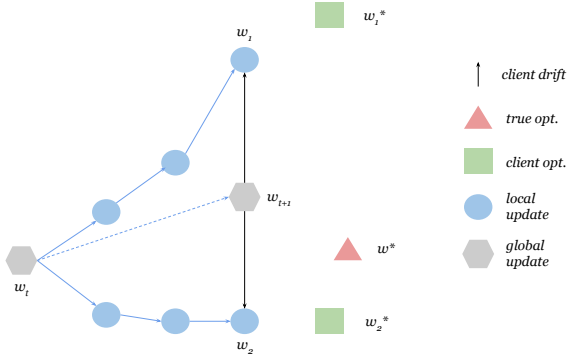
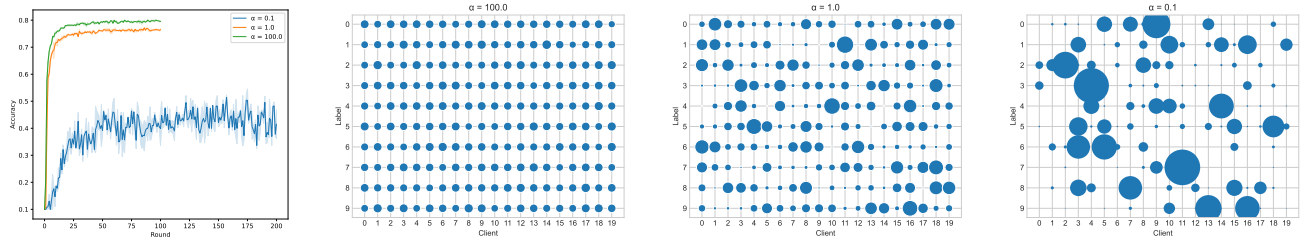


Figure 2: Client drift phenomenon in FedAvg is depicted for two learners performing three local updates (light blue circles). Such local steps move towards the individual client optima w_i^* (green square) by minimizing their cost function for local data. The server aggregates client updates by average, and therefore the updated version of the global model at the next round (i.e., w_{t+1}) (gray hexagon) move towards the average of client optima instead of to the true optimum w^* (red triangle) [26].

It should be emphasized that client drift becomes more pronounced when there are many local updates, which can occur due to a high number of epochs or a small local batch size. This amplifies the divergence among personalized client models. However, using too few local iterations can result in high cumulative communication costs, meaning that a high number of rounds is required to achieve global model convergence. This has been observed in various studies, such as [26, 27, 28].

For the sake of clarity, Fig. 3a reports some qualitative results to highlight the degradation, in terms of global model generalization, introduced by non-IIDness. The accuracy



(a) FedAvg's accuracy. (b) $\alpha = 100.0$ (c) $\alpha = 1.0$ (d) $\alpha = 0.1$

Figure 3: Figure (a) illustrates how the accuracy of the globally trained model using FedAvg decreases as the data heterogeneity increases. A lower α coefficient indicates a higher level of data heterogeneity. The experiments conducted to generate this figure utilized the CIFAR-10 dataset and a ResNet-18 architecture. Figures (b), (c) and (d) represent the label distribution (labels ranging from 0 to 9) present in the private data of 20 clients that were used to generate the results shown in Figure (a). The label distribution is depicted for three different values of the concentration parameter α, which governs the Dirichlet distribution, as described in [29]. This is an example of label distribution skew, simulated with distribution-based label imbalance.

of a global model trained while considering three different levels of data heterogeneity (low, moderate, and high, visually depicted in Fig. 3b, c, d, respectively) is tracked, empirically showing that a higher level of data heterogeneity translates into the reduced performance of the FedAvg global model (test accuracy). The data among clients is allocated following a distribution-based label imbalance [20], and ruled by a concentration hyperparameter α (the larger the α the higher the heterogeneity level) [29].

Recent studies have demonstrated that the predictions of FedAvg's global models are significantly inconsistent across consecutive communication rounds [30, 31, 32]. At each round, the model focuses on a subset of the data and struggles to generalize to the previously seen ones. This is particularly evident when the global model exhibits a lower accuracy in predicting some classes with respect to its previous-round version. Such an observation indicates that global knowledge is prone to be forgotten, and this phenomenon has been associated with similar behaviors in related research fields, such as *catastrophic forgetting* in continual learning and *catastrophic interference* in multitask learning research.

Furthermore, according to Caldarola et al., the geometry of the loss surface may provide insight into the global model's ability to generalize, where flatter minima tend to correspond to better generalization [31]. Caldarola et al. also suggests that global models trained using FedAvg typically converge towards sharp minima, a phenomenon that is exacerbated when the data is heterogeneous [31]. From an orthogonal perspective, weighted averaging as a method for global parameter aggregation is less detrimental for clients with flat optima compared to those with sharp local minima, since minor changes in parameters can cause a significant increase in error in the latter case [33, 34].

Addressing the aforementioned shortcomings, that is enhancing the generalization ability of FedAvg's global, a plethora of solutions have been proposed in literature. In the following, we review, characterize and classify the state-of-the-art methods to tackle the detrimental effects caused by heterogeneous data.

2.3. Generalization vs Personalization Ability

Canonical FL aims to learn a single global model that can generalize well to unseen data that follow the distribution of the combined local datasets from all clients, denoted as $\mathcal{D} = \bigcup_{k \in K} \mathcal{D}_k$. As a consequence, the empirical performance of FL methods that enhance generalization is evaluated on a server-side test set, drawn from the same distribution of $\mathcal{D}$ [12]. It is worth noting that generalization methods do not aim to optimize the global model for each client, since clients are likely to hold heterogeneous data among each other and with respect to $\mathcal{D}$.

An orthogonal line of research, i.e., personalized FL, aims to learn personalized local models that are customized to each client. Although local models could be trained without federation, clients hold a limited amount of data, which does not allow them to learn general-purpose reliable models (e.g., the client models risk overfitting local data). Hence, the primary scope of personalized FL is to build a global model, prone to adapt to local data distributions. Different from methods for generalization, personalized FL approaches are empirically evaluated on local indicators. Usually, clients are supposed to possess a train set and a test/evaluation set drawn from the same data distribution (i.e., clients' data are randomly split between train and test/evaluation partitions according to practitioner-defined criteria) [35]. The performance of the proposed personalization method is evaluated on clients' test/evaluation set after fine-tuning the local model for a fixed amount of epochs and, then, reporting aggregated performance among clients (e.g., the mean and standard deviation of clients' loss and accuracy on test/evaluation set) [14, 15]. As already stated, we decided to focus this specific paper exclusively on FedAvg-like methods that aim to improve the models' generalization ability.

3. Enhancing the Generalization Ability of the Global Model

For the sake of clarity, as shown in Fig. 4, we primarily differentiate the considered solutions between client and server-side methods. The first class of algorithms modifies

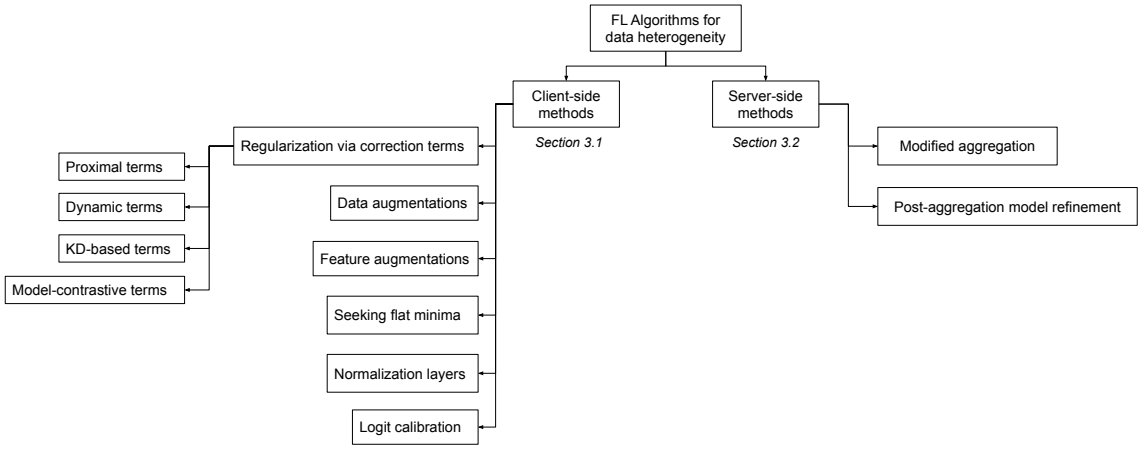


Figure 4: Schematic classification of approaches proposed in literature to enhance global model generalization in presence of heterogeneous data.

the local training routine of clients - with respect to FedAvg - to explicitly limit the client drift phenomenon. At the other end of the spectrum, server-side optimizations modify or enhance the aggregation step of FedAvg to limit the degradation introduced by averaging drifted client models. Usually, client-side and server-side methods are orthogonal and can be used in combination to amplify the performance gain (e.g., to reduce the total rounds needed to reach global model convergence or to achieve better global model generalization).

In the experimental part of the work, we provide the researchers and practitioners in the field with an original technical contribution of empirically evaluating and comparing a selection of algorithms, among the ones presented here. The comparative analysis is the basis for a technical and practical discussion on the possible advantages and weaknesses of the considered solutions when implemented and deployed to support various verticals in real deployment environments. Furthermore, our code is publicly available for result reproducibility and to foster research advances.

Typically, the generalization ability of the global model is measured via metrics, such as loss and accuracy on a test set. The test set contains unseen examples uniformly distributed among classes, drawn from the same distribution of $\mathcal{D}$, i.e. the combination of local distributions. The reviewed solutions, unless differently specified, adopt a star-shaped topology (client-server as in FedAvg) and proceed in synchronous rounds; usually, classification tasks are considered for design and evaluation.

3.1. Client-side Methods

This section goes through solutions that intervene in the local training phase so to limit the degradation introduced by client model drift (see Fig. 2) on the global model performance since local models tend to diverge from the received global model and from the model of other clients in the federation in presence of heterogeneous data.

In their pioneering study [65], the authors show via an experimental analysis that FedAvg's test accuracy can be considerably improved in the presence of heterogeneous data by supplementing participants' private datasets with a small portion of the globally shared data provided by the server. While this approach is effective, it comes at the expense of reduced decentralization and necessitates transmitting the common data to the learners.

Without resorting to data-sharing techniques, a common approach to limit the client drift phenomenon is to locally regularize the training phase by introducing one or more additional terms to the client's objective function. This typically translates into applying a penalty that scales with some measure of the client drift, controlling the divergence from the current global model. As discussed in the following, most of the works leverage the current global model as a reference for local training, using it as an anchor for not letting the local model parameters drift apart (e.g., [27]) or using it as a guide during local training (e.g., [45, 48, 50, 51] [46]), or again employing it to prevent *catastrophic forgetting* [32]. Other lines of work do not actively leverage the global model in their solutions, for example focusing on normalization layers (e.g., [41]), or seeking flatter minima in local objective functions [31, 54, 34], or again proposing federated adaptations of traditional data augmentation techniques [37, 52], or modifying the output *softmax* layer and/or the cross-entropy loss before the network output (e.g., [42, 53]). Authors of [22] also provide theoretical convergence analysis, and observe the need for a decaying learning rate at client-side in non-IID settings.

3.1.1. Local Regularizations via Correction Terms

Specifically, FedProx [27] adds a proximal term to the local objective function of clients that is proportional to the *L2-norm* distance between the received global model parameters and the in-learning local parameters, i.e.:

$$\mathcal{L}_{Prox}(w, w_t) = \mathcal{L}(w) + \frac{\mu}{2}||w - w_t||^2 \tag{2}$$

where $\mathcal{L}$ is the classic supervised loss for the classification task, μ weights the impact of the L2-norm regularization

Table 1
Classification of reviewed client-side methods, listed by year (from oldest to most recent).

Method	Control Term(s)	Normalization Methods	Logit Calibration	Flat Minima	Feature Aug.	Data Aug.
FedProx [27]	✓					
SCAFFOLD [26]	✓					
FedDANE [36]	✓					
FedMix [37]						✓
SiloBN [38]		✓				
MOON [39]	✓					
FedDyn [40]	✓					
FedBN [41]		✓				
FedRS [42]			✓			
AdaBest [43]	✓					
FedGen [44]						✓
FedGKD [45]	✓					
FedCodl [46]	✓					
FedNTD [32]	✓					
FedMAX[47]	✓					
FedCAD [48]	✓					
DMFL [49]			✓			
FedSSD [50]	✓					
FedMLB [51]	✓					
FedAlign [52]	✓					
HarmoFL [34]				✓		
FedLC [53]			✓			
FedSAM [31]				✓		
FedSAM [54]				✓		
FedASAM [31]				✓		
MoFedSAM [54]				✓		
FedFA [55]					✓	
FixBN [56]		✓				
FedTAN [57]		✓				
WSM [31]			✓			

Table 2
Classification of reviewed server-side methods, listed by year (from oldest to most recent).

	Modified Aggregation	Post-Aggregation Refinement
FedAvgM [29]	✓	
FedNova [28]	✓	
FedBE [58]		✓
FedDF [59]		✓
FedOpt [22]	✓	
FedAux [60]		✓
FedZKT [61]		✓
FedDNA [62]	✓	
SWA [31]	✓	
FedFTG [63]		✓
GMA [64]	✓	

term, w and w_t are respectively the in-learning local model and the current global model. Besides reducing the impact of heterogeneous data, FedProx was also designed to enable uneven amounts of local training iterations among clients.

Similarly to FedProx, FedDyn [40] proposes a dynamic tuning that is adapted based on the current global model. However, while FedProx does not result in aligning local and global stationary points, during each round, FedDyn dynamically tailors the local objective function with a penalty term so that the local optima is asymptotically consistent with stationary points of the global optima. More concretely, the penalty term in FedDyn is composed of a linear term and a quadratic penalty term, and the resulting objective function is in the form of:

$$\mathcal{L}_{Dyn}(w, w_t, w_{t-1}^k) = \mathcal{L}(w) - \langle \nabla \mathcal{L}(w_{t-1}^k), w \rangle + \frac{\alpha_{dyn}}{2} ||w - w_t||^2 \tag{3}$$

where $\nabla \mathcal{L}(w_t^k) = \nabla \mathcal{L}(w_{t-1}^k) - \alpha_{dyn}(w_t^k - w_t)$ and it is computed recursively, α is a hyperparameter, and clients are indexed by k. [1] It is worth noting that FedDyn is treated here

[1] $\nabla \mathcal{L}(w_0^k)$ is initialized to 0.

Algorithm 2: FedDyn.

Server executes:
  initialize w_1
  $\nabla\mathcal{L}(w_0^k) \leftarrow 0$
  for each round $t = 1, 2, 3, ..$
    $c \leftarrow max(C \times K, 1)$
    $S_t \leftarrow$ (random set of c clients)
    for each client $k \in S_t$ **in parallel**
      $w_t^k \leftarrow w_t$
      $w_t^k \leftarrow \arg\min_w$ Eq. 3
      $\nabla\mathcal{L}(w_t^k) \leftarrow \nabla\mathcal{L}(w_{t-1}^k) - \alpha_{dyn}(w_t^k - w_t)$
    // FedDyn needs a server-side state h_t
    $h_t \leftarrow h_{t-1} - \alpha_{dyn}\frac{1}{K}(\sum_{k\in S_t} w_t^k - w_t)$
    $w_{t+1} \leftarrow (\frac{1}{|S_t|}\sum_{k\in S_t} w_t^k) - \frac{1}{\alpha_{dyn}}h_t$

as a client-side optimization, but it also requires a server-side state, and can be used in combination with other client-side optimizations (e.g., FedDyn plus local multi-branch regularization, FedMLB, as presented in the experimental results of [51]). Algorithm 2 displays the FedDyn method. While rooted in theory, FedDyn requires clients to maintain state (i.e., h_t in Algorithm 2) throughout the process. Furthermore, as demonstrated in [43], FedDyn needs a high rate of client re-sampling to exhibit a stable convergence, otherwise, the unbounded increase of h_t norm will cause instability in the learning process.

Conceptually similar, SCAFFOLD [26] leverages correction terms (i.e., *control variates*) for local gradients to ensure that the local update moves towards the true optimum. Such correction terms are used as proxy for the clients' true gradients, whose disclosure would require unsustainable frequency of synchronization (after each local step, clients should communicate local gradients). Instead, during each FL round, control variates are locally computed and updated, then communicated back to the server for aggregation at the end of local training, and finally aggregated to produce global control variates which are broadcast during the next round. Hence, SCAFFOLD doubles the communication cost with respect to FedAvg.

In addition, it is worth noting that SCAFFOLD and FedDyn require prior knowledge of the number of total clients in order to properly weight their accumulator (i.e., h_t in Algorithm 2), which may be an unrealistic assumption in large-scale FL settings. The authors of [43] propose Adabest, which can be seen as a generalization of FedDyn that does not assume prior knowledge about the number of participating clients and does not require a high re-sampling rate of clients to ensure stability. As a result, AdaBest is most beneficial in the presence of large-scale settings with low participation rates.

FedDANE [36] is a method inspired by DANE [66] and its inexact variant [67], which combines the proximal term used in FedProx with a gradient correction term similar to SCAFFOLD. The update process involves two steps: compute the gradient correction term and solve the Newton-type sub-problem inexactly, the locally computed gradients of the local objective functions are collected and then averaged to approximate the full gradients. To keep from gathering all the locally-computed gradients, FedDANE approximates the complete gradients by aggregating the gradients of a random subset of learners. It is important to note that FedDANE doubles the communication cost with respect to FedAvg, as previously noted for SCAFFOLD. Furthermore, despite being rooted in theory, FedDANE exhibits inferior empirical performance compared to FedAvg and FedProx.

FedMAX [47] focuses on limiting the divergence in *activation vectors*, i.e. input to the neural network's classification layer, among clients and aims at making the activation vectors of same classes as similar as possible in the federation. FedMAX employs a correction term in the loss function of clients to maximize the entropy of such activation vectors, so to limit their uncontrolled divergence.

3.1.2. Local-Global Knowledge Distillation

Besides proximal or dynamic regularizations, also regularization techniques based on Knowledge Distillation (KD) [68] can be employed to tackle the client drift phenomenon by locally distilling global knowledge. Instead of directly considering global and local model parameters as in FedProx, the global model is seen as a teacher in a distillation framework [69], and its model responses on private samples are used to guide the local learning and limit the divergence of client models [45, 32, 48, 50, 51] [46].

FedGKD [45], FedCodl [46], and FedNTD [32] regularize the local objective function with a distillation-based term as follows:

$$\mathcal{L}_{KD}(w, w_t) = \gamma\mathcal{L}(w) + \tau^2(1-\gamma)\mathcal{D}(q_w^\tau, q_{w_t}^\tau). \quad (4)$$

The q^τ terms represent the model output after a softmax scaled with a temperature τ, which regulates the smoothness of the model predictions[2]. q_w^τ and $q_{w_t}^\tau$ are the smoothed predictions of the updated local model and the smoothed predictions of the global model, respectively. γ weights the impact of the KD-based term. The distance function $\mathcal{D}(\cdot)$ can be computed, for example, via the Kullback-Leibler (KL) divergence. FedGKD [45] also proposes to consider the mean of the last M historical models as teacher. With $M = 1$, i.e., considering the current global model as the local teacher, FedGKD collapses to FedCodl [46]. Differently, FedNTD [32] computes q_w^τ and $q_{w_t}^\tau$ by excluding the logit associated to the true label.

However, only leveraging the distance among teacher-student outputs does not necessarily transfer the ability of the global model to produce better intermediate features with respect to client models. FedMLB, presented in [51], extends the KD-based framework presented in FedGKD, FedNTD, and FedCodl, by considering hybrid paths composed of

[2]The scaled softmax outputs a tensor q^τ, composed by elements j, computed as $q^\tau(j) = \frac{exp(x(j)/\tau)}{\sum_i exp(x(i)/\tau)}$, with x being the logits before softmax.

client model blocks followed by global model blocks, encouraging the local model to reproduce intermediate features similar to the ones generated by the global model. The model block is intended as an ensemble of consecutive layers, e.g. a residual block in a residual architecture. However, while FedGKD, FedNTD and FedCodl introduce a negligible computation overhead on clients, FedMLB needs to propagate gradients through the hybrid paths, introducing significant computation overhead and increased local training time. Also, FedGKD, FedNTD, FedCodl and FedMLB require the global model to be stored (and not overwritten as it happens in FedAvg) during local training, hence doubling the memory requirements. This limitation can, in practice, be avoided for FedGKD, FedNTD and FedCodl by firstly computing the predictions of the received global models, and then proceeding with local training, and overwriting the global model.

Considering the local-global distillation framework, the authors in [48] observe that imitating the output of an inaccurate global model as a teacher can misguide local training (for example, in early rounds the global model exhibits poor predictions). To address this issue, FedCAD [48] introduces a class-wise adaptive weight to control the impact of distillation, so that the global model knowledge is distilled when accurate. FedCAD computes the class-wise adaptive weight based on the performance of the global model on a proxy dataset and broadcasts this matrix along with model parameters at each communication round. FedSSD [50] builds on FedCAD by incorporating the credibility of the global model at the instance level when computing the distillation term in local training.

3.1.3. Model-contrastive Learning

MOON [39], inspired by typical data-level contrastive learning frameworks such as SimCLR [70], is a model-level contrastive learning method designed to rectify the local training of clients. While typical contrastive learning compares representations of different augmented views computed on the same image (*positive pairs*) and representations of different augmented versions computed on different images (*negative pairs*), in MOON [39], positive and negative pairs come from the representations computed by the global model and the previous-version local model on the same private example, respectively. The key idea is to correct the local training of individual clients by maximizing the consensus on the representation learned by the current client model and the representation learned by the server model. At the same time, MOON encourages the representation learned by the current client model to diverge from the representation learned by the previous-version client model (client model resulting from the last active round). [3] In practice, this translates to a local objective function composed of a typical term to account for supervised loss (e.g., cross-entropy loss among ground-truth labels and predicted soft labels) and a model-contrastive term. It is worth noting that, differently from FedAvg, MOON implies stateful clients, demanding to store the last trained local model. Additionally, the approach requires the modification of the locally learned model by adding a projection head, also adding computation and storage overhead attributed to the need of producing and comparing the representation of three models (global, current, and last-version local model).

3.1.4. Data and Feature Augmentation

From another perspective, FedMix [37] adapts the simple, well-known $Mixup$ augmentation [71] to the federated context. Similarly, the authors of [52] recently demonstrated that augmentations like GradAug [72] and StochDepth [73] significantly improve the performance of FedAvg in presence of heterogeneous data, though they introduce substantially computation overhead. Addressing the issue, the authors of [52] design a method, FedAlign, which has similar effects and performance as GradAug in FL but with a reduced computation overhead. In particular, FedAlign aligns the Lipschitz constants (i.e. top Hessian eigenvalues) of deeper layers of the neural network – most prone to overfit client distribution [74] – through the use of slimmed sub-blocks and a distillation-based regularization term. FedGen [44] employs a generator model, broadcast by the server to clients round by round, to be used to locally obtain augmented training examples, while incorporating global knowledge as inductive biases. To train the generator, FedGen requires the disclosure of the local label count, besides local model parameters. While approaches such as FedMix and FedGen works at the input level, FedFA [55] proposes to augment features by injecting Gaussian noise calibrated according to the channel-wise statistics (i.e., mean and standard deviation) of the participants in the federation.[4] In this way, FedFA allows client models to be trained over data examples drawn from more variegated feature distributions, encouraging participant-invariant representation learning, and eventually producing a server model that generalizes better.

3.1.5. Seeking Flat Minima

Recently, the technique of Sharpness-Aware Minimization (SAM) [75] has been employed and proposed in FedSAM [31, 54]. In fact, the authors of [31] suggest that the geometry of the loss surface is a relevant indicator for generalization, and in particular that sharp minima translate to a lack of generalization. To this end, FedSAM locally encourages flatter minima and smoother loss landscape by leveraging SAM, and resulting in improved performance of global model. Also, adaptive and momentum versions of FedSAM have been proposed to speed up convergence, respectively named FedASAM [31], based on Adapative SAM [76], and MoFedSAM [54], which leverages a momentum parameter.

Similarly inspired, HarmoFL [34] promotes client models that are easy to aggregate via weight perturbation (as

[3] To produce the representation, MOON [39] injects a projection head before the output layer of the model. The considered representation is the output of the projection head.

[4] FedFA [55] explicitly focuses on data in the form of images.

highlighted in Sec. 2.2, clients that exhibit flatter loss landscape are more amenable to be aggregated via FedAvg's averaging scheme). In addition, HarmoFL couples the weight-perturbation strategy with amplitude normalization of local training. To this scope, HarmoFL leverages special layers in the client's model architecture which perform amplitude normalization and collect local amplitudes, to be then aggregated by the server to produce and employ a global one.

3.1.6. Normalization Methods

As demonstrated in several works [77, 41, 56, 78, 57, 56], in presence of non-IID data, Batch Normalization (BN) layers [79] are detrimental for global model performance due to a mismatch among local statistics which cause shifted local features learned by clients. FedBN [41] proposes to only locally train BN's parameters (batch variance, batch mean, scale, and shift parameters) and never communicate the updates of such layers to the server for aggregation. Similarly, SiloBN [38] keeps batch variance and mean local, but communicates the scale parameter and the shift parameters for global merge. FixBN [56] proposes to regularly train, communicate and aggregate all the BN's parameters, but freezing BN's global and local statistics from a certain communication round onward. The work in [77] and the work in [78], respectively, suggest employing Group Normalization [80] or Layer Normalization [81], which do not use mini-batch statistics, in place of BN layers. FedTAN [57] tailors the local training procedure of FedAvg to ensure convergence properties in presence of BN layers at the cost of several intra-round synchronizations among client and servers.

3.1.7. Logit Calibration

A parallel line of work focuses on mitigating the detrimental effect caused by label distribution skew, a specific type of data heterogeneity, on the classification layer of neural networks. The recent work in [53] and in [49] respectively propose FedLC and DMFL that use a modified cross-entropy loss (*fine-grained calibrated cross-entropy*) which calibrates the local model output (i.e., logits), before feeding them to the softmax output layer, according to per-class local number of samples. A similar approach has been recently proposed in [30] with *re-weighted softmax* (WSM) to reduce catastrophic forgetting. Similarly inspired, FedRS [42] advocates the use of a *restricted softmax*, which employs rescaling factors to account for under-represented classes on the local dataset.

3.2. Server-side Methods

This section proposes an updated and in-depth overview of server-side approaches proposed to reduce the degradation introduced by averaging drifted model parameters/updates. Relevant lines of work include proposing different weighting schemes with respect to FedAvg, rectifying the aggregated global model with an additional phase (e.g., a distillation phase), or interpreting the model updates as "pseudo-gradient" and using an optimizer of choice – usually different from SGD – to apply them to the global model.

3.2.1. Modified Aggregation Procedure

FedNova [28] improves FedAvg in the aggregation step by considering that different clients may perform a different number of local steps (i.e., the number of mini-batches in the local training) during each round. This can result, for example, from clients holding different amounts of local examples given a fixed batch size, or from learners having approximately the same amount of local examples but different amounts of local epochs. To ensure that the global updates are not biased, FedNova designs a normalized averaging method to replace the simple FedAvg update, i.e, before updating the global model, it scales the client updates based on the local number of local steps.

Conceptually similar to the works discussed in Section 3.1.6, FedDNA [62] revisits the aggregation of statistical parameters belonging to normalization layers in the neural network (e.g., batch mean and batch variance of BN layers). In particular, FedDNA aggregates statistical parameters with an importance weighting method, and they are optimized collaboratively leveraging an adversarial learning approach, while gradient-based parameters are updated via regular FedAvg's scheme.

From another perspective, the authors of [31] suggest incorporating stochastic weight averaging (SWA) from [82] into the aggregation stage of FedAvg so to enhance the resilience of the learning process [31]. The work shows that the SWA technique is most effective when applied toward the end of the training phase, specifically after completing 75% of the total rounds.

Another body of research leverages server-side optimizers which handle aggregated model updates as "pseudo-gradient". In fact, in FedAvg, instead of uploading the brand-new model parameters, clients typically send updates to the server, which are computed as the difference between the received global model and the locally computed client model, i.e. the update rule of FedAvg can be written as:

$$w_{t+1} = \frac{1}{|S_t|}\sum_{i\in S_t} w_t^k = w_t + \frac{1}{|S|}\sum_{i\in S_t}(w_t^k - w_t) \quad (5)$$

with S_t being the subset of clients selected for round t. For the sake of clarity, in Eq. 5 the model parameters/updates are not weighted according to the number of local data samples, but it does not have an impact on the following observation: defining the client updates as $\Delta_t^k := w_t^k - w_t$ and their aggregated form as $\Delta_t := \frac{1}{|S_t|}\sum_{i\in S_t}\Delta_t^k$, we have:

$$w_{t+1} = w_t + \Delta_t = w_t - (-\Delta_t) \quad (6)$$

Hence, the server update rule in FedAvg is equivalent to applying SGD to the "pseudo-gradient" $-\Delta_t$ with learning rate $\eta_s = 1$. From this observation, FedAvg can be considered a specialization of a more general framework, called FedOpt, that uses SGD as a server-side optimizer [22].

Algorithm 3 presents the FedOpt framework and variants. This body of work shows that optimizers different from

SGD can be used on the server, leading to the proposal of using server-side momentum or adaptive optimizers.[5] It is worth noting that, with the introduction of a server-side optimizer, η_s represents the global learning rate, which is an additional hyperparameter to tune. For server-side momentum, FedAvgM [29] uses SGD with momentum at the server side, while FedAdagrad, FedYogi, and FedAdam are the specializations of FedOpt, employing Adagrad [83, 84], Yogi [85] or Adam [86], respectively. It is worth noting that FedAvgM, FedAdam, FedYogi, and FedAdagrad do not introduce computation or communication overhead on clients with respect to FedAvg. The algorithms exhibit improved convergence rate both theoretically and empirically, also in presence of heterogeneous data. Algorithm 4 describes FedAdagrad, FedYogi and FedAdam.

Drawing connections with *out-of-distribution* generalization, Tenison *et al*. designed a Gradient Masked Aggregation (GMA) as an alternative for regular FedAvg's update aggregation [64]. In GMA, a soft mask, based on sign agreement among client updates, is applied to the aggregated updates. It is worth noting that GMA can be plugged into any other algorithm as an alternative to FedAvg aggregation (e.g., into FedOpt framework).

3.2.2. *Post-Aggregation Refinement*

Concluding, other contributions focus on fine-tuning the global model after aggregation, before a new FL round begins. In particular, KD-based approaches have been proposed to rectify the server model by applying post-aggregation ensemble distillation. In practice, the global model is fine-tuned by mimicking the aggregated predictions of clients' models on common data. This class of solutions supposes the existence of publicly-available proxy data (semantically similar to the clients' local dataset) or leverages model generators. In detail, FedDF [59] builds the ensemble of clients' model outputs by computing their average, while FedBE [58] employs a Bayesian model ensemble to merge client predictions to improve aggregation robustness. FedAUX [60] further extends FedDF's procedure by utilizing unsupervised pre-training on auxiliary data to determine a suitable model initialization for local feature extractor. FedAUX also weighs the ensemble predictions on the proxy data based on a (ϵ, δ)-differentially private certainty score [87] of each participant model. While server-side ensemble distillation methods assume the availability of a (semantically similar) proxy dataset, FedFTG [63] conducts server-side rectification of the global model through data-free knowledge distillation, where the server adversarially trains both a generator model and the global model, and refines the latter on synthetic data. A similar data-free refinement strategy is proposed in FedZKT [61].

[5] The client-side optimizer (ClientOpt in Algorithm 3) is supposed to be SGD.

Algorithm 3: FedOpt.

Server executes:
  initialize w_0
  for each round $t = 0, 1, 2, 3, ..$
    $c \leftarrow max(C \times K, 1)$
    $S_t \leftarrow$ (random set of c clients)
    for each client $k \in S_t$ **in parallel**
      $\Delta_t^k \leftarrow$ **ClientOpt**(k, w_t)
    $\Delta_t \leftarrow \sum_{i \in S_t} \frac{n_k}{n} \Delta_t^k$
    $w_{t+1} \leftarrow$ **ServerOpt**$(w_t, -\Delta_t, \eta_s, t)$

Algorithm 4: FedOpt: **FedAdagrad**, **FedYogi** and **FedAdam**.

Server executes:
  initialize w_0
  initialize decay parameters $\beta_1, \beta_2 \in [0, 1)$, $v_{-1} \geq \tau^2$
  for each round $t = 0, 1, 2, 3, ..$
    $c \leftarrow max(C \times K, 1)$
    $S_t \leftarrow$ (random set of c clients)
    for each client $k \in S_t$ **in parallel**
      $\Delta_t^k \leftarrow$ ClientUpdate(k, w_t)
    $\Delta_t \leftarrow \sum_{i \in S_t} \frac{n_k}{n} \Delta_t^k$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
    $v_t \leftarrow v_{t-1} + \Delta_t^2$ **(FedAdagrad)**
    $v_t \leftarrow v_{t-1} - (1 - \beta_2)\Delta_t^2 sign(v_{t-1} - \Delta_t^2)$ **(FedYogi)**
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ **(FedAdam)**
    $w_{t+1} \leftarrow w_t + \eta_s \frac{m_t}{\sqrt{v_t} + \tau}$

ClientUpdate(k, w_t)
  $w \leftarrow w_t$
  $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B)
  for each local epoch e from 1 to E
    for batch $b \in \mathcal{B}$
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  $\Delta \leftarrow w - w_t$
  return Δ to server

4. Empirical Evaluation

In this section, we propose an experimental evaluation of the discussed approaches, considering both client- and server-side classes. Further contributing to the current body of knowledge, we propose a benchmark of hybrid FL approaches, emerging as a combination of client- and server-side algorithmic tweaks, discussing the tradeoffs that emerge.

4.1. Experimental Setup

In the following, we provide some details on the dataset used in the analysis, the partitioning strategy used to produce

different levels of heterogeneity, along with the experimental parameters comprising the evaluation matrix. All the experimental simulations have been run in a Python virtual environment on a machine equipped with two NVIDIA RTX A5000 GPUs.

4.1.1. Datasets and baselines

We conducted a set of experiments on the CIFAR-100 dataset, which consists of 60.000 examples of 32x32 color images - 50.000 for training and 10.000 for testing - belonging to 100 classes. We partitioned the training set to simulate 100 clients in the federation. As standard in FL algorithm evaluations, we used the method proposed in [29] to generate the clients' shard of data. A concentration parameter α tunes the heterogeneity of local training sets. With a high α value (e.g., > 100) the distribution of labels among clients tends to be homogeneous while lowering the α value translates to local data with different amounts of examples per label until the extreme case of local examples belonging to only one label.

We performed simulations with two levels of data heterogeneity, determined by $\alpha = 0.3$ and $\alpha = 0.1$. In the first setting ($\alpha = 0.3$), we restrict the clients to have a balanced local set (i.e., the same amount of total examples) and we distribute the examples as in [51], where the authors provide the exact list of examples per client. In the latter case ($\alpha = 0.1$), we do not restrict the learners to have the same amount of total data points, and learners can have significantly unbalanced data sets. When possible, different clients do not rely on repeated examples. Fig. 5a and Fig. 5b depict the distribution of labels among the clients with varying α values.

We consider FedAvg as a baseline and benchmark a selection of state-of-the-art FL techniques among the ones presented in the first part of this paper. For client-side optimizations, we compared the following methods: FedProx, FedNTD, FedGKD (M=1), FedGKD (M=5), FedMLB, FedDyn, FedSAM, FedFA[6]. For server-side optimizations, we considered FedAvgM and FedAdam. Also, we coupled promising compatible approaches, i.e. FedAdam + FedMLB, FedAvgM + FedMLB, FedDyn + FedMLB, FedAvgM + FedSAM, FedAdam + FedSAM, FedAvgM + FedFA, FedAdam + FedFA, to assess the gain deriving from both client- and server-side optimizations. While preserving the general nature of our study and without loss of generality, we have excluded from the analysis works that either assume the existence of (semantically-similar) additional public datasets or approaches known to be more bandwidth-hungry in the upload link when compared to FedAvg or techniques that do not disclose their code base and/or did not include the hyperparameter tuning procedure, or that do not use standard classification tasks in their evaluation.

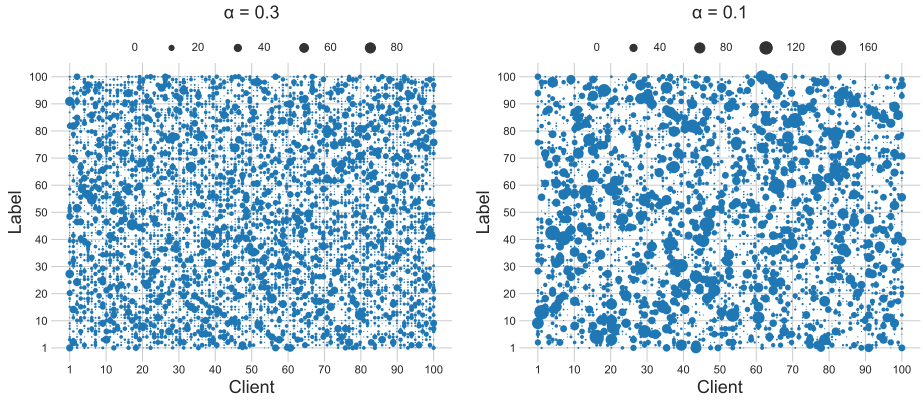
[6]FedGKD (M=1) [45] and FedCodl [46] are equivalent when setting the temperature τ to 1.

4.1.2. Implementation details

Similar to [40, 51], in all the experiments, SGD with a learning rate fixed to 0.1 is used as local optimizer, and the global learning rate is set to 1.0, except for FedAdam, which used 0.01 for both local and global learning rate, and for FedAdam + FedSAM, which used 0.1 for local and 0.01 for global learning rate. Local epochs are fixed to 5 and a random fraction of 0.05 (5 %) clients are selected per round. A weight decay with a factor of 0.001 is applied to limit local overfitting. Local batch size is determined so that each client performs 50 local updates. Gradient clipping is performed to stabilize local training. The local learning rate is exponentially decayed with a factor of 0.998 similar to the work in [40, 51]. The model architecture used in our experiments is ResNet-18 [88], but we have replaced the batch normalization layer with group normalization as suggested in [77, 51]. We used random rotation, horizontal flip, and random crop as preprocessing layers. For a fair comparison, seeds are used to select a set of random clients at each round, to perform local data preprocessing, and to initialize client models; different algorithms are compared using the same set of seeds on more runs. For ease of readability, we report a distilled set of parameters in Tab.3.

For algorithmic-specific hyperparameters, we proceeded as follows:

- For FedProx we tuned μ in $\{0.1, 0.01, 0.001\}$. μ controls the weight of the proximal term in the local objective function. We selected $\mu = 0.01$ for $\alpha = 0.3$ and $\mu = 0.001$ for $\alpha = 0.1$.
- For FedGKD we set γ to 0.2, as in the original paper. γ controls the weight of the KD-based term in the local objective function.
- For FedNTD we selected λ in $\{0.3, 1.0\}$.
- For FedDyn we set α_{dyn} equal to 0.1 as in the original paper.
- For FedMLB λ_1 and λ_2 are both set to 1 (λ_1, λ_2 weight the impact of the hybrid cross-entropy loss and the KD-based loss). 5 blocks are considered, formed as in the original paper, where conv1, conv2_x, conv3_x, conv4_x, conv5_x and the fully connected layer constitutes a single block.
- For FedSAM we tuned ρ in {0.5, 0.1, 0.05, 0.01} and we selected 0.05 as best value. ρ defines the neighborhood size for seeking flat minima.
- For FedFA we augment the clients' ResNet18 architecture with five FA layers, one after each ResNet block. The server aggregates the FedFA's channel-wise feature statistics and uses a regular ResNet18 architecture for test. FA layers are tuned as in the original paper [55].
- For FedAvgM we tuned the momentum parameter among $\{0.4, 0.6, 0.8, 0.9\}$, and selected 0.6 for $\alpha = 0.1$ and 0.8 for $\alpha = 0.3$.



(a) $\alpha = 0.3$, balanced datasets. (b) $\alpha = 0.1$, unbalanced datasets.

Figure 5: Label distribution with different levels of heterogeneity. (a) Distribution of labels (0-99) on clients (100 clients) with concentration parameter set to 0.3. Local datasets have the same cardinality. This is the exact same setting of [51]. (b) Distribution of labels (0-99) on clients (100 clients) with concentration parameter set to 0.1.

Table 3
Hyperparameter tuning of evaluated algorithms.

Method	Selected ($\alpha = 0.3$, bal.)	($\alpha = 0.1$, unbal.)
FedProx	$\mu = 0.01$	$\mu = 0.001$
FedNTD	$\lambda = 0.3$	
FedGKD (M=1)	$\gamma = 0.2$	
FedGKD (M=5)	$\gamma = 0.2$	
FedDyn	$\alpha_{dyn} = 0.1$	
FedMLB	$\lambda_1 = 1$, $\lambda_2 = 1$	
FedSAM	$\rho = 0.05$	
FedAdam	$\tau = 0.001$, β_1=0.9, β_2=0.999	
FedAvgM	momentum = 0.8	momentum = 0.6

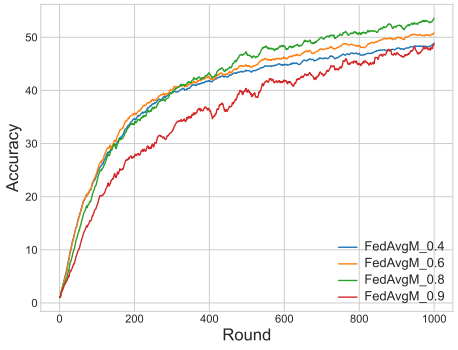


Figure 6: FedAvgM accuracy with different momentum parameters, $\alpha = 0.3$.

- For FedAdam we set τ (a constant for numerical stability) equal to 0.001 as in [51, 22].

Table 3 reports the per-algorithm hyperparameter selection.

The algorithms are implemented in Python using the TensorFlow library. It is worth noting that most of the code available for such algorithms is implemented leveraging the Pytorch library, hence the accompanying code of this paper constitutes a valuable contribution to the FL community and is readily made available in [1]. The code base also provides the implementation of MOON, but the latter is not included in the comparison analysis since it locally modifies the model architecture by injecting a 2-layer head.

4.2. Experimental Results

Table 4, Tab.5 and Tab. 6 respectively report the results of the client-side, server-side and hybrid FL methods, for different levels of data heterogeneity. The arrows denote whether the higher (accuracy) or the lower (rounds) is better. Since both final accuracy and convergence speed are relevant indicators for federated learning [89], we report the performance attained at different rounds and the number of needed communication rounds to reach a target accuracy. For the solutions that do not achieve the target accuracy in 1000 rounds, we use the notation 1000+.

Table 7 reports the local execution time of the best performing client-side approaches (i.e., FedDyn, FedMLB, FedSAM, FedFA) with respect to FedAvg. The reported time (in seconds) is the average local training time across 5 clients per round, over 150 rounds, considering the setting ($\alpha = 0.3$, balanced), 500 local examples, 5 local epochs, and 50 local updates. The per-round pseudo-random selection of clients is ruled by the same set of seeds for all the measurements to ensure fairness. Clients run sequentially in our centralized simulations. We use the results in Tab. 7 as a proxy for qualitatively assessing the computation overhead, with respect to FedAvg, introduced by the considered state-of-the-art algorithms.

4.2.1. Client-side Methods

Figure 7a and Fig. 7f show the test accuracy for the considered client-side methods. As it is shown, FedProx has a similar performance to FedAvg, while being very sensitive to its additional hyperparameter (reported best results with $\mu = 0.01$ for $\alpha = 0.3$ and with $\mu = 0.001$ for $\alpha = 0.3$) since

Table 4
Comparison among considered client-side methods on CIFAR-100 within different federated learning settings. The accuracy at the target round and the number of communication rounds to reach the target test accuracy are based on smoothed average with parameter 0.9.

Method	$\alpha = 0.3$ (balanced)					$\alpha = 0.1$ (unbalanced)			
	Accuracy (%, ↑)		Rounds (↓)			Accuracy (%, ↑)		Rounds (↓)	
	500R	1000R	25%	45%	47%	500R	1000R	25%	39%
FedAvg	42.78	47.02	127	756	965	33.06	39.83	309	889
FedProx	42.36	46.91	128	754	1000+	33.21	39.83	309	889
FedNTD	42.38	46.66	123	764	1000+	33.91	40.67	307	807
FedGKD (M=1)	43.13	47.64	122	657	935	33.74	40.56	309	806
FedGKD (M=5)	42.61	46.48	123	663	1000+	34.07	40.55	307	802
FedDyn	48.90	58.11	89	334	414	36.39	48.02	269	536
FedMLB	49.21	55.60	122	390	440	31.97	42.68	368	743
FedSAM	43.62	48.54	143	575	729	34.12	42.83	321	708
FedFA	46.54	52.08	149	443	548	37.48	47.17	310	549

Table 5
Comparison among considered server-side methods on CIFAR-100 in two different federated learning settings. The accuracy of the target round and the number of communication rounds to reach the target test accuracy are based on smoothed average with parameter 0.9.

Method	$\alpha = 0.3$ (balanced)					$\alpha = 0.1$ (unbalanced)			
	Accuracy (%, ↑)		Rounds (↓)			Accuracy (%, ↑)		Rounds (↓)	
	500R	1000R	25%	45%	55%	500R	1000R	25%	39%
FedAvg	42.78	47.02	127	756	1000+	33.06	39.83	309	889
FedAvgM	42.99	53.55	113	464	1000+	35.38	42.51	250	724
FedAdam	44.56	53.09	139	422	1000+	41.76	47.42	193	430

with larger μ values the local model is more constrained to remain close to the global model's weights and, in turn, it struggles to adapt to local data. Furthermore, it is worth noting that FedProx has been also designed to handle a different number of local computations (e.g., different amounts of local epochs), which is not the setting we reproduced (all the clients perform the same number of local updates in our experiments).

One interesting result is that the baseline methods that use the global model output in a KD-based framework (FedGKD, FedNTD) only show negligible gains or even degrade global the model accuracy. As also claimed in [48], this can be explained by the fact that the predictions of

Table 6
Comparison among hybrid methods on CIFAR-100 in two different federated learning settings. The accuracy at the target round and the number of communication rounds to reach the target test accuracy are based on smoothed average with parameter 0.9.

Method	$\alpha = 0.3$ (balanced)					$\alpha = 0.1$ (unbalanced)			
	Accuracy (%, ↑)		Rounds (↓)			Accuracy (%, ↑)		Rounds (↓)	
	500R	1000R	25%	45%	55%	500R	1000R	25%	39%
FedAvg	42.78	47.02	127	756	1000+	33.06	39.83	309	889
FedAvgM + FedFA	48.56	55.25	113	329	849	**41.94**	49.71	**221**	**413**
FedAvgM + FedSAM	49.43	56.75	122	330	751	39.01	47.27	235	504
FedAdam + FedSAM	49.58	56.55	144	265	790	38.88	49.65	279	501
FedAvgM + FedMLB	52.56	58.55	133	333	636	35.30	50.21	329	603
FedAdam + FedMLB	51.22	58.17	124	357	742	39.6	50.74	242	476
FedDyn + FedMLB	**58.97**	**63.31**	**81**	**222**	**264**	34.92	**54.59**	341	573

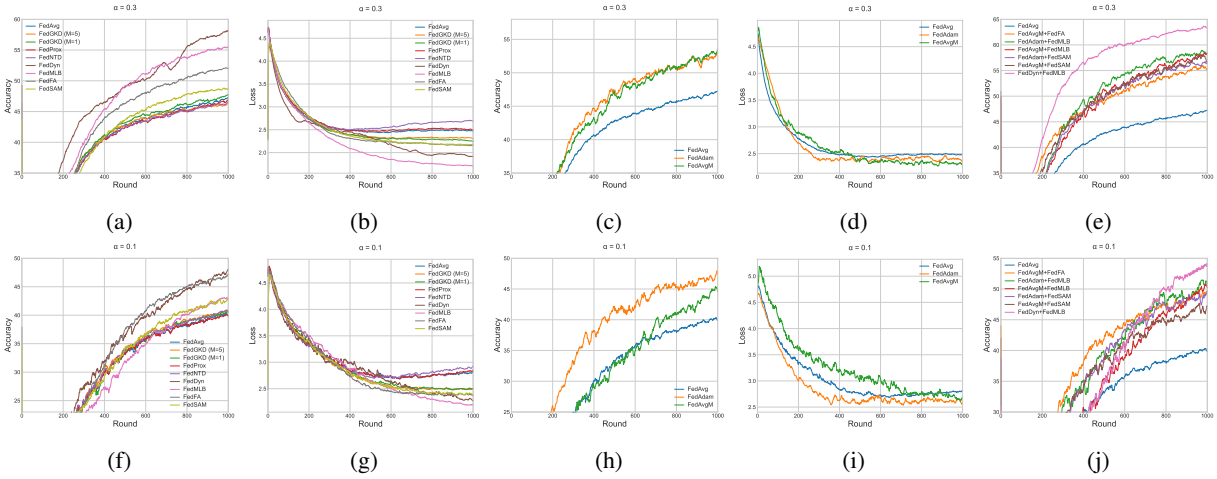


Figure 7: The figure depicts the global model's performance (accuracy and loss), also reported in Tab. 4, Tab. 5 and Tab. 6, on test data across 1000 communication rounds. The first row of charts refers to the setting ($\alpha = 0.3$, balanced datasets), the second one to ($\alpha = 0.1$, unbalanced datasets). (a, f) Accuracy of considered client-side methods with different levels of data heterogeneity. (b, g) Loss of considered client-side methods with different level of data heterogeneity. (c, h) Accuracy of considered server-side methods with different level of data heterogeneity. (d, i) Loss of considered server-side methods with different level of data heterogeneity. (e, j) Accuracy of considered combined methods with different level of data heterogeneity.

Table 7
Execution time (in seconds) of best performing client-side algorithms and overhead with respect to FedAvg, setting ($\alpha = 0.3$, balanced).

Method	Time (s, ↓)	wrt FedAvg (↓)
FedAvg	10.91	
FedDyn	11.07	≈
FedFA	15.91	1.45x
FedSAM	17.64	1.61x
FedMLB	25.25	2.31x

the global model can be misleading during training, especially in the first communication rounds, due to inaccurate predictions. Hence, mimicking the response of the global model is suboptimal and can hamper local training. Contrarily, FedMLB is more robust in presence of heterogeneous data and largely outperforms the other client-side KD-based methods, even though it results to be more effective on the setting ($\alpha = 0.3$, balanced) with a relative increase of 18% with respect to FedAvg which drops to slightly more than 7% for ($\alpha = 0.1$, balanced). This result confirms that local-global distillation is motivated: the guidance of the global model can be significantly beneficial to tackle data heterogeneity. At the same time, in response-based local-global distillation (FedGKD, FedNTD), locally learned intermediate features may drift apart, and approaches that also pose constraints to intermediate representations are more effective. From a different perspective, the global model should be better at producing general intermediate features, and its influence on local training is beneficial.

However, FedMLB introduces significant local computation and memory overhead, as well as more local execution time (see Tab. 7), which obviously translates to an increase in energy consumption. While not representing an issue in cross-silo federations, such overhead may preclude the participation of resource-constrained nodes in cross-device settings.

It is worth mentioning that FedGKD (with M = 5 historical models) shows a slight improvement over FedAvg in the $\alpha = 0.1$ setting with limited memory and computation overhead. However, it requires 2 times the server-client communication costs compared to FedAvg and the other methods, since the server also transmits the aggregated historical model. Figure 7b and Fig. 7g depict the global model loss on test data for client-side approaches, and the trends that emerge are quite consistent with the analysis conducted on global model accuracy. An interesting aspect is that FedGKD reaches lower loss values with respect to FedAvg, which however translates either to a slight improvement or even in degradation in accuracy. Similarly, FedMLB results in lower test loss with respect to FedDyn, which conversely achieves the best accuracy, in both the heterogeneity levels.

FedFA exhibits significant improvements in generalization over FedAvg, at negligible cost of memory overhead and limited overhead of local execution time – FedFA resulted to be the solution with the lowest execution time after FedDyn (see Table 7). It is worth highlighting that FedFA requires to modify the local model architecture (FedFA leverages special layers that inject feature noise) and requires disclosing additional local information about aggregated statistics of data – which have negligible communication cost.

FedSAM results to be more effective in presence of higher heterogeneity levels ($\alpha = 0.1$, unbalanced), surpassing FedMLB in global model accuracy both at 500R and 1000R. Contextually, FedSAM requires longer local training time with respect to FedDyn and FedFA which

however exhibit better generalization, while not requiring prior knowledge about the federation or modifications to the local model architecture.

With respect to client-side approaches, FedDyn shows impressive results in both the considered settings and achieves the best accuracy after 1000 learning rounds (respectively 58.11% and 48.02%), while introducing marginal overhead in local execution time (see Table 7). Recall that FedDyn is classified among client-side approaches, even though it maintains a server-side state which is vital to the algorithm. Its memory requirements, however, are at least doubled with respect to FedAvg to be attributed to the computation of its penalty term; furthermore, FedDyn supposes a prior knowledge about the number of total clients, which may be not a realistic assumption in certain circumstances.

4.2.2. Server-side Methods

When considering the server-side approaches - FedAvgM and FedAdam - they exhibit significant improvements over FedAvg in both the considered settings (see Tab. 5, Fig. 7c, Fig. 7h). The global model loss on test data confirms the trends of global model accuracy (see Fig. 7d, 7i). Overall, FedAdam achieves the best results, performing slightly worse than FedAvgM in ($\alpha = 0.3$, balanced) but showing a significant improvement in the more extreme setting. Also, FedAdam is consistently faster in achieving target accuracy levels. We note once again that, differently from client-side methods, FedAvgM and FedAdam do not introduce client-side overhead. Hwever, they require additional hyperparameters to be tuned, i.e. the server-side learning rate and other hyperparameters specific to the optimizer (e.g., momentum for FedAvgM). Although their best configurations outperform FedAvg, they seem quite sensitive to such hyperparameters, as also shown in [22]. For the sake of clarity, in Fig. 6, we report the performance of FedAvgM with different momentum parameters.

4.2.3. Hybrid Methods

Table 6 reports the results of hybrid methods, derived as a combination of the most promising client- and server-ide approacheswhich are: FedAvgM + FedMLB, FedAdam + FedMLB, FedDyn + FedMLB, FedAvgM + FedSAM, FedAdam + FedSAM, and FedAvgM + FedFA. Figure 7e and Fig. 7j depict the achieved accuracy of the considered combined solutions.

The first two methods exhibit very similar performance, achieving approximate levels of accuracy, while the Adam-based one converges faster. Overall, the combination FedDyn + FedMLB attains the best generalization performances. For the setting ($\alpha = 0.3$, balanced), FedDyn + FedMLB results to be the best method in convergence speed and in maximum accuracy achieving 55% accuracy in 264 communication rounds, with FedAvg never achieving a comparable accuracy even in 1000 communication rounds. For ($\alpha = 0.1$, unbalanced), FedDyn + FedMLB converges slower than other solutions but shows the best final accuracy. It is worth noting that, while outperforming all the other considered solutions, FedDyn + FedMLB requires the combined overhead of the two solutions, resulting in a non-negligible increase in memory usage and local execution time for client devices (see Tab. 7), hence at the expense of greater energy consumption, as well as supposing FedDyn's prior knowledge about the federation. FedAvgM + FedFA shows an impressive slope in the first part of the training, resulting as the faster method to achieve the target levels of accuracy (25%, 39%) in the most extreme setting. FedSAM seems to be most effective when coupled with server-side optimization, bridging the gap with other client-side methods.

4.2.4. Discussion

The analysis shows that improving FedAvg's robustness to data heterogeneity is a challenging task. Superior approaches, e.g. FedDyn or FedMLB, usually introduce additional overhead (memory, execution time and energy consumption, communication costs), assume prior domain knowledge or leverage proxy datasets. Overall, the results confirm that (1) approaches rooted in theory, like FedDyn, are extremely effective, (2) transferring knowledge between global and local models, in an effort to limit catastrophic forgetting, effectively improves generalization, with utilizing intermediate features being crucial, (3) seeking for flatter local minima also translates to improved robustness of the global model confirming that SAM-trained local models are more amenable to averaged aggregation, and (4) feature-augmentation layers can be an additional architectural element to be considered when designing neural networks for use in federated environments. Finally, it is evident the benefit of combining client- and server-side techniques so to boost the effectiveness of single-sided approaches.

5. Open Problems and Future Directions

In this section, we summarize a list of open problems that emerge from the literature, and we provide our vision on near term future directions of this field.

5.1. Evaluation on Real Testbeds

Almost all the approaches proposed in literature evaluate the proposals on simulated federation of clients, running them in a centralized manner and leveraging datacenter-like hardware, such as powerful GPUs. However, often such evaluations do not take into account the actual possible overhead on client devices, which is a very relevant aspect for cross-device settings, as argued throughout in this document. Usually, the performance analysis only focuses on metrics such as the global model generalization (or personalization) ability, the accuracy and loss on test data either from a local or global distribution, comparing them with respect to well-known baselines, e.g. FedAvg for synchronous, star-shaped FL. We claim that evaluations of state-of-the-art solutions should include empirical assessment of the client overhead, including execution time, computational cost and energy consumption, measured on hardware representative of real (far)edge devices.

Table 8
The table qualitatively summarizes the local *computation*, *communication*, and local *memory* overhead introduced by each surveyed work if compared with the canonical FedAvg approach. The symbol = means no overhead, ≈ means negligible overhead, ↑ means moderate overhead, and ↑↑ means significant overhead. The column *Additional Information* reports whether the method requires or discloses additional information about FedAvg (other than model weights and model updates). The column *Proxy Data* reports whether the specific work requires the existence of publicly available proxy data. FedOpt Family stands for FedAvgM, FedAdam, FedAdagrad and FedYogi. The column *Stateful* indicates whether clients should maintain a state over rounds.

	Work	Computation	Communication	Memory	Additional Information	Proxy Data	Stateful
Client Side	FedProx	≈	=	↑	No	No	No
	FedNTD [32]	≈	=	↑	No	No	No
	FedCodl [46]	≈	=	↑	No	No	No
	FedGKD (M=1) [45]	≈	=	↑	No	No	No
	FedGKD (M>1)	≈	↑↑	↑	No	No	No
	FedDyn [40]	≈	=	≈	No	No	Yes
	FedMLB [51]	↑↑	=	↑↑	No	No	No
	FedSAM [31, 54]	↑	=	↑	No	No	No
	FedFA [55]	↑	=	↑	Yes	No	No
	SCAFFOLD [26]	≈	↑↑	≈	No	No	No
	FedMix [37]	≈	=	≈	No	No	No
	MOON [39]	↑	≈	↑	No	No	No
	FedBN [41]	=	≈	=	No	No	No
	AdaBest [43]	≈	=	↑	No	No	Yes
	FedGen [44]	↑	≈	↑	Yes	No	No
	FedMAX[47]	≈	=	=	No	No	No
	FedCAD [48]	≈	=	↑	No	Yes	No
	FedSSD [50]	≈	=	↑	No	Yes	No
	FedAlign [52]	↑	=	≈	No	No	No
	HarmoFL [34]	↑	=	↑	Yes	No	No
	FedASAM [31]	↑	=	↑	No	No	No
	MoFedSAM [54]	↑	=	↑	No	No	No
	SiloBN [38]	=	≈	=	No	No	No
	FixBN [56]	=	≈	=	No	No	No
	FedTAN [57]	≈	↑	=	No	No	No
	FedRS [42]	≈	=	=	No	No	No
	FedLC [53]	≈	=	=	No	No	No
	DMFL [49]	≈	=	=	No	No	No
	WSM [31]	≈	=	=	No	No	No
Server Side	FedOpt Family [22, 29]	=	=	=	No	No	No
	FedNova [28]	=	=	=	No	No	No
	FedBE [58]	=	=	=	No	Yes	No
	FedDF [59]	=	=	=	No	Yes	No
	FedAux [60]	=	=	=	No	Yes	No
	FedZKT [61]	=	=	=	No	No	No
	FedDNA [62]	=	=	=	No	No	No
	SWA [31]	=	=	=	No	No	No
	FedFTG [63]	=	=	=	Yes	No	No
	GMA [64]	=	=	=	No	No	No

5.2. Hyperparameter Tuning

Almost all the algorithms treated here, either acting client- or server side, introduce additional hyperparameters to be tuned. Furthermore, in real FL settings, the level of data heterogeneity is unknown *a priori*, hence algorithms that perform well with the same tuning across different levels of non-IIDness are generally preferable. However, while in traditional centralized training it is acceptable to apply, e.g., grid search for hyperparameter tuning, such practice may be not feasible for cross-silo settings (e.g., federations of health institutions) and surely unfeasible with cross-device settings (e.g., federation of edge devices). In light of this, it is reasonable to expect a growing interest in the design of easy-to-tune and auto-tuning algorithms for data-heterogeneous FL.

5.3. Considering and Quantifying Additional Trade-off

Table 8 provides a summary of the overhead and requirements that the surveyed works introduce over the baseline FedAvg. As also highlighted in [69] and throughout Sec. 3, a subset of works needs to disclose additional information (beside model parameters and model updates). For example, FedFA [55] discloses additional feature-based statistics, or FedGen [44] needs to transmit the local label count, just to mention a few. However, the possible privacy breach (e.g., private data reconstruction) implied by exchanging additional information is often unclear or left unexplored. Some of the surveyed solutions (e.g., MOON [39]) require clients to maintain a state throughout the FL process, while this could be difficult to meet when clients have a low rate of participation or are likely to participate only once. Orthogonally, other contributions from the literature design methods that suppose the existence of publicly available proxy data, which should be semantically similar to the clients' data. Such an assumption may be hardly satisfied in real-world scenarios when employing generative models usually requires clients to disclose additional information. We believe that the above introduced overhead and requirements have been undervalued in the current literature, where the accent is almost exclusively on improving the global model performance (e.g., its accuracy).

5.4. Homogeneous Benchmarking

Ensuring homogeneous benchmarking and result reproducibility in FL research is still an open issue. In fact, even though a de-facto standard practice is to evaluate novel algorithms on regular datasets partitioned among clients via distribution-based label imbalance, usually following the method proposed in [29], each novel work being proposed defines its own federated setting as a combination of several degrees of freedom, such as the hyperparameter for the Dirichlet distribution in the partition method of [29], having balanced or unbalanced client datasets, the number of total learners, the participation rate of clients, just to mention a few. Furthermore, the model architecture chosen for a specific dataset, and the relative hyperparameters (local epochs, local batch size, local and global learning rate, and others) represent additional elements that can influence results. Reaching a consensus on the minimum experimental setting that should be provided when assessing the superiority of a novel method would be beneficial for the field, as well as speeding up the evaluation phase for researchers who could easily replicate a standard experimental setup.

6. Conclusive Remarks

In this work, we provided a detailed review of state-of-the-art algorithms proposed to limit the degradation in generalization ability caused by data heterogeneity among clients in FL settings. The reviewed approaches are grouped into two macro-categories, i.e., client and server-side methods, further refined to finer subcategories, highlighting their core technical features. A comprehensive empirical assessment was conducted employing different (simulated) levels of data heterogeneity on CIFAR-100 with ResNet-18, shedding some light on the tradeoffs that emerge. The analysis shows that some solutions are more effective than others, but these benefits may come at an unsustainable cost for client devices (e.g., overhead in execution time and energy consumption) or assuming unrealistic prior knowledge about the federation.

Acknowledgements

The authors wish to express their gratitude to the anonymous reviewers for their constructive comments and suggestions. This work was partially supported by the European Union's NextGenerationEU instrument, under the Italian National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3, enlarged partnership "Telecommunications of the Future" (PE0000001), program "RESTART."

References

[1] A. Mora, "Federated learning algorithms with heterogeneous data distributions," https://github.com/alessiomora/fl_algorithms_non_iid_2, 2023.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] Q. Li and other, "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347–3366, 2023.

[4] P. Bellavista, L. Foschini, and A. Mora, "Decentralised Learning in Federated Deployment Environments: A System-Level Survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 1, pp. 1–38, 2021.

[5] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *arXiv preprint arXiv:1912.04977*, 2019.

[6] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[7] E. Union, "Complete guide to general data protection regulation compliance," URL https://gdpr.eu/.

[8] C. Mazzocca, N. Romandini, M. Colajanni, and R. Montanari, "Framh: A federated learning risk-based authorization middleware for healthcare," *IEEE Transactions on Computational Social Systems*, 2022.

[9] U. D. of Health and H. Services, "The hipaa privacy rule," URL https://www.hhs.gov/hipaa/for-professionals/privacy/index.html, accessed on May 2020.

[10] Cisco, "Cisco global cloud index: Forecast and methodology, 2016–2021 white paper," URL https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_white-paper-c11-738085.pdf, accessed on April 2020.

[11] X. Qiu, T. Parcollet, D. Beutel, T. Topal, A. Mathur, and N. Lane, "Can federated learning save the planet?" in *NeurIPS-Tackling Climate Change with Machine Learning*, 2020.

[12] H. B. McMahan *et al.*, "Communication-efficient Learning of Deep Networks from Decentralized Data," *arXiv preprint arXiv:1602.05629*, 2016.

[13] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[14] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Towards enhanced representation for federated image classification," *arXiv preprint arXiv:2106.06042*, 2021.

[15] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=g0a-XYjpQ7r

[16] C. Xie, O. Koyejo, and I. Gupta, "Asynchronous federated optimization."

[17] E. Baccarelli, M. Scarpiniti, A. Momenzadeh, and S. S. Ahrabi, "Afafed—asynchronous fair adaptive federated learning for iot stream applications," *Computer Communications*, vol. 195, pp. 376–402, 2022.

[18] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[19] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022.

[20] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated Learning on Non-iid Data Silos: An Experimental Study," in *Proc. of IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[21] A. Mora, D. Fantini, and P. Bellavista, "Federated Learning Algorithms with Heterogeneous Data Distributions: An Empirical Evaluation," in *Proc. of IEEE/ACM SEC*. IEEE, 2022, pp. 336–341.

[22] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[23] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *International Conference on Learning Representations*, 2019.

[24] M. Chen, B. Mao, and T. Ma, "Fedsa: A staleness-aware asynchronous federated learning algorithm with non-iid data," *Future Generation Computer Systems*, vol. 120, pp. 1–12, 2021.

[25] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–16.

[26] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for on-device federated learning," *arXiv preprint arXiv:1910.06378*, 2019.

[27] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[28] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[29] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[30] G. Legate, L. Caccia, and E. Belilovsky, "Re-weighted softmax cross-entropy to control forgetting in federated learning," *arXiv preprint arXiv:2304.05260*, 2023.

[31] D. Caldarola, B. Caputo, and M. Ciccone, "Improving Generalization in Federated Learning by Seeking Flat Minima," in *Proc. of European Computer Vision Conference*. Springer, 2022, pp. 654–672.

[32] G. Lee, M. Jeong, Y. Shin, S. Bae, and S.-Y. Yun, "Preservation of the global knowledge by not-true distillation in federated learning," in *Advances in Neural Information Processing Systems*, 2022.

[33] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[34] M. Jiang, Z. Wang, and Q. Dou, "Harmofl: Harmonizing Local and Global Drifts in Federated Learning on Heterogeneous Medical Images," in *Proc. of AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 1087–1095.

[35] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," *arXiv preprint arXiv:1910.10252*, 2019.

[36] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Feddane: A federated newton-type method," *arXiv preprint arXiv:2001.01920*, 2020.

[37] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "FedMix: Approximation of Mixup under Mean Augmented Federated Learning," in *Proc. of International Conference on Learning Representations*, 2020.

[38] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed Federated Learning for Multi-centric Histopathology Datasets," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*. Springer, 2020, pp. 129–139.

[39] Q. Li, B. He, and D. Song, "Model-contrastive Federated Learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 713–10 722.

[40] D. A. E. Acar, Z. Yue, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated Learning Based on Dynamic Regularization," *arXiv preprint arXiv:2111.04263*, 2021.

[41] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated Learning on Non-iid Features via Local Batch Normalization," *arXiv preprint arXiv:2102.07623*, 2021.

[42] X.-C. Li and D.-C. Zhan, "Fedrs: Federated Learning with Restricted Softmax for Label Distribution Non-iid Data," in *Proc. of ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 995–1005.

[43] F. Varno, M. Saghayi, L. Rafiee, S. Gupta, S. Matwin, and M. Havaei, "Minimizing client drift in federated learning via adaptive bias estimation," *arXiv preprint arXiv:2204.13170*, 2022.

[44] Z. Zhu, J. Hong, and J. Zhou, "Data-free Knowledge Distillation for Heterogeneous Federated Learning," in *Proc. of International Conference on Machine Learning*. PMLR, 2021, pp. 12 878–12 889.

[45] D. Yao, W. Pan, Y. Dai, Y. Wan, X. Ding, H. Jin, Z. Xu, and L. Sun, "Local-global knowledge distillation in heterogeneous federated learning with non-iid data," *arXiv preprint arXiv:2107.00051*, 2021.

[46] X. Ni, X. Shen, and H. Zhao, "Federated optimization via knowledge codistillation," *Expert Systems with Applications*, vol. 191, p. 116310, 2022.

[47] W. Chen, K. Bhardwaj, and R. Marculescu, "Fedmax: Mitigating Activation Divergence for Accurate and Communication-efficient Federated Learning," in *Proc. of Machine Learning and Knowledge Discovery in Databases: European Conference*. Springer, 2021, pp. 348–363.

[48] Y. He, Y. Chen, X. Yang, Y. Zhang, and B. Zeng, "Class-wise adaptive self distillation for heterogeneous federated learning," 2022.

[49] X. Ran, L. Ge, and L. Zhong, "Dynamic margin for federated learning with imbalanced data," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[50] Y. He, Y. Chen, X. Yang, H. Yu, Y.-H. Huang, and Y. Gu, "Learning critically: Selective self-distillation in federated learning on non-iid data," *IEEE Transactions on Big Data*, 2022.

[51] J. Kim, G. Kim, and B. Han, "Multi-level branched regularization for federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 11 058–11 073.

[52] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, "Local Learning Matters: Rethinking Data Heterogeneity in Federated Learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8397–8406.

[53] J. Zhang, Z. Li, B. Li, J. Xu, S. Wu, S. Ding, and C. Wu, "Federated Learning with Label Distribution Skew via Logits Calibration," in *Proc. of International Conference on Machine Learning*. PMLR, 2022, pp. 26 311–26 329.

[54] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu, "Generalized Federated Learning via Sharpness Aware Minimization," in *Proc. of International Conference on Machine Learning*. PMLR, 2022, pp. 18 250–18 280.

[55] T. Zhou and E. Konukoglu, "Fedfa: Federated feature augmentation," *arXiv preprint arXiv:2301.12995*, 2023.

[56] J. Zhong, H.-Y. Chen, and W.-L. Chao, "Making batch normalization great in federated deep learning," *arXiv preprint arXiv:2303.06530*,

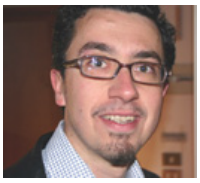
2023.
[57] Y. Wang, Q. Shi, and T.-H. Chang, "Why batch normalization damage federated learning on non-iid data?" *arXiv preprint arXiv:2301.02982*, 2023.
[58] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," *arXiv preprint arXiv:2009.01974*, 2020.
[59] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.
[60] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, "Fedaux: Leveraging unlabeled auxiliary data in federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
[61] L. Zhang and X. Yuan, "Fedzkt: Zero-shot knowledge transfer towards heterogeneous on-device models in federated learning," *arXiv preprint arXiv:2109.03775*, 2021.
[62] J.-H. Duan, W. Li, and S. Lu, "FedDNA: Federated Learning with Decoupled Normalization-layer Aggregation for Non-iid Data," in *Proc. of Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 722–737.
[63] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning Global Model via Data-free Knowledge Distillation for Non-iid Federated Learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 174–10 183.
[64] I. Tenison, S. A. Sreeramadas, V. Mugunthan, E. Oyallon, E. Belilovsky, and I. Rish, "Gradient masked averaging for federated learning," *arXiv preprint arXiv:2201.11986*, 2022.
[65] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.
[66] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning*, 2014, pp. 1000–1008.
[67] S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola, "Aide: Fast and communication efficient distributed optimization," *arXiv preprint arXiv:1608.06879*, 2016.
[68] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[69] A. Mora, I. Tenison, P. Bellavista, and I. Rish, "Knowledge distillation for federated learning: a practical guide," *arXiv preprint arXiv:2211.04742*, 2022.
[70] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," in *Proc. of International Conference on Machine Learning*. PMLR, 2020, pp. 1597–1607.
[71] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.
[72] T. Yang, S. Zhu, and C. Chen, "Gradaug: A new regularization method for deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 207–14 218, 2020.
[73] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep Networks with Stochastic Depth," in *Proc. of European Conference on Computer Vision*. Springer, 2016, pp. 646–661.
[74] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.
[75] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware Minimization for Efficiently Improving Generalization," in *Proc. of International Conference on Learning Representations*, 2021.
[76] J. Kwon, J. Kim, H. Park, and I. K. Choi, "Asam: Adaptive Sharpness-aware Minimization for Scale-invariant Learning of Deep Neural Networks," in *Proc. of International Conference on Machine Learning*. PMLR, 2021, pp. 5905–5914.
[77] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The Non-iid Data Quagmire of Decentralized Machine Learning," in *Proc. of International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.
[78] Z. Du *et al.*, "Rethinking Normalization Methods in Federated Learning," in *Proc. of the 3rd International Workshop on Distributed Machine Learning*, 2022, pp. 16–22.
[79] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. of the International Conference on Machine Learning*. pmlr, 2015, pp. 448–456.
[80] Y. Wu and K. He, "Group Normalization," in *Proc. of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
[81] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[82] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.
[83] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," *arXiv preprint arXiv:1002.4908*, 2010.
[84] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.
[85] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Advances in neural information processing systems*, 2018, pp. 9793–9803.
[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[87] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
[88] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[89] M. Al-Shedivat, J. Gillenwater, E. Xing, and A. Rostamizadeh, "Federated learning via posterior averaging: A new perspective and practical algorithms," *arXiv preprint arXiv:2010.05273*, 2020.

CRediT authorship contribution statement

Alessio Mora: Conceptualization, Methodology, Software, Data curation and visualization, Writing - Original draft preparation. **Armir Bujari:** Conceptualization, Methodology, Writing. **Paolo Bellavista:** Conceptualization, Methodology, Writing.

Alessio Mora (Student Member, IEEE) received a MS degree in Computer Engineering at the University of Bologna. He is currently a Ph.D. candidate in Computer Science and Engineering, at the University of Bologna. His research interests include Decentralized Learning, with particular focus on Federated Learning, Deep Learning and Internet of Things.

Armir Bujari (IEEE Member) received the Ph.D. degree in Computer Science from the University of Bologna, Italy, in 2014. He is currently an Associate Professor of Computer Science and Engineering at the University of Bologna, Italy. His research interests are primarily focused on edge/fog computing applications in the industrial domain, next-generation architectures, networks and services.



Paolo Bellavista (Senior Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2001, where he is currently a Full Professor His research interests include middleware for mobile computing, QoS management in the cloud continuum, infrastructures for big data processing in industrial environments, and performance optimization in wide-scale and latency-sensitive deployment environments. He served as the Scientific Coordinator for the H2020 IoTwins Project. He serves on the Editorial Boards for IEEE Communications Surveys and Tutorials, IEEE Transactions on Network and Service Management, IEEE Transactions on Service Computing, ACM CSUR, ACM TIOT, and PMC (Elsevier).