Client-Side Optimisation Strategies for Communication-Efficient Federated Learning

Jed Mills, Jia Hu, Geyong Min

***Abstract*—Federated Learning (FL) is a swiftly evolving field within machine learning for collaboratively training models at the network edge in a privacy-preserving fashion, without training data leaving the devices where it was generated. The privacy-preserving nature of FL shows great promise for applications with sensitive data such as healthcare, finance, and social media. However, there are barriers to real-world FL at the wireless network edge, stemming from massive wireless parallelism and the high communication costs of model transmission. The communication cost of FL is heavily impacted by the heterogeneous distribution of data across clients, and some cutting-edge works attempt to address this problem using novel client-side optimisation strategies. In this paper, we provide a tutorial on model training in FL, and survey the recent developments in client-side optimisation and how they relate to the communication properties of FL. We then perform a set of comparison experiments on a representative subset of these strategies, gaining insights into their communication-convergence tradeoffs. Finally, we highlight challenges to client-side optimisation and provide suggestions for future developments for FL at the wireless edge.**

***Index Terms*—Federated Learning, Communication-efficiency, Edge Computing, Deep Learning, Optimisation.**

I. Introduction

FEDERATED Learning (FL) is an emerging paradigm within Machine Learning (ML) that uses distributed computation to preserve data-privacy. In ML, models are trained on datasets to achieve good generalisation performance (e.g., for image classification or sentiment analysis). Larger datasets typically lead to better generalisation. Traditionally, when organisations wish to train a model using data generated by a group of clients (e.g., mobility models from smartphone users, or environmental models in wireless sensing), the client data is collected and the organisation trains the model in a datacentre.

However, due to growing public concern regarding data-privacy and the establishment of associated laws (such as the General Data Protection Regulation in the EU), data-owners are rapidly becoming less willing or able to share their data. FL was conceived for training useful ML models from client data, without that data leaving the devices where it was generated.

One of the major real-world settings for FL is the 'cross-device' scenario, where many edge devices (usually connected wirelessly, for example over cellular networks) collaborate to train the model [1]. Typically, the edge devices perform rounds of local computation (model training on their private data), and are coordinated by a centralised server with which they communicate. This scenario is characterised by a large number of clients with unreliable and limited-bandwidth connections, where each client possesses a small fraction of the total federated dataset. As real-world FL involves communicating large quantities of data (usually in the form of model parameters) over these connections for long deployments (up to weeks at a time), communication between the edge clients and the central server represents a major bottleneck. Therefore, despite the huge potential to address privacy-preserving ML, there exist substantial challenges to designing useful FL systems for the network edge. Chief among these are:

- **Huge wireless parallelism:** cross-device FL can host millions of wireless clients performing distributed computation, with high-latency and low-bandwidth connections to the coordinating server (compared to datacentre training) and high unreliability due to their wireless connectivity, leading to long training times. The ML pipeline typically involves several iterations of hyperparameter tuning and model redesign, meaning FL's long training times are highly impractical. Therefore, efficient algorithms and reliable wireless protocols are required to make real-world FL practicable.
- **Heterogeneous client data:** the data stored on clients can be very diverse stemming from their different behaviour and interactions with the environment. The total data across all devices is therefore partitioned in a non-Independent and Identically Distributed (non-IID) fashion. Non-IID client data has been extensively shown to harm the convergence speed and final performance of the FL model [1]–[3], further increasing the communication burden and training time.
- **Large communication costs:** Deep Neural Networks (DNNs) are extremely popular ML models due to their performance and wide range of applications. The overarching trend in DNN development has been towards larger and deeper models, with state-of-the-art DNNs reaching billions of parameters. Most FL algorithms involve transmitting these parameters thousands of times between clients and server, placing a large burden on the communication networks connecting them.

There has been a flurry of research activity aiming to address the above challenges from a variety of angles. The authors of [4] survey FL from a wireless-communications perspective, covering communication-reduction strategies as well as wireless-FL use-cases. [5] provides a broad survey of FL in regards to networking and communications classified by the FL-related challenge that each work addresses, including statistical and communication-efficiency challenges. [1]

J. Mills, J. Hu and G. Min are with the Computer Science Department, University of Exeter, UK. Email: {jm729, j.hu, g.min}@exeter.ac.uk. Corresponding authors: Jia Hu, Geyong Min.

presents an extensive background to the FL scenario, covering many works related to algorithmic, privacy and systems-design developments.

However, none of these surveys specifically cover novel client-side optimisation techniques and their implications for communications. These techniques alter the way that the model is optimised on client devices, and have a large impact on the performance and communication properties of the FL system. None of these techniques are mentioned in [4], while [5] only cites two relevant algorithms, and [1] misses many of the cutting-edge developments in this sub-field. Our major contributions are therefore:

- We provide the first survey focussing on client-side optimisation strategies and their relations to the communications properties of FL.
- We discuss the trade-offs between communication costs, hyperparameter selection, and convergence inherent to these strategies, which are important factors in wireless-edge FL.
- We compare the performance of a representative subset of the surveyed strategies, obtaining insights into their benefits to communication costs and convergence speed on two benchmark FL datasets.
- We shed light on future research directions within this topic, including for novel strategy combinations, design of relevant datasets, realistic wireless-edge testing, and algorithms that better leverage the network properties of the edge.

The rest of this article is organised as follows: Section II describes FL at the wireless edge and the challenges posed by heterogeneous client data and communication bottlenecks, Section III surveys recent developments in client-side optimisation strategies, Section IV provides an experimental comparison between a representative subset of these strategies, and Section V concludes the paper and provides suggestions for future research directions.

II. The Federated Learning Scenario

In this section, we describe the wireless-edge FL scenario, and discuss the impact of heterogeneous client data on the communication-efficiency of FL.

A. Client-Server Federated Learning

FL clients each possess a small fraction of the total federated dataset. These clients collaborate to train a model that has better performance than the models that would be created from independent training. Clients are typically low-powered wireless-edge equipment such as smartphones, sensors, or embedded devices. Coordination of the clients is usually performed by a central server at the network core.

Most FL algorithms operate in rounds consisting of model distribution, client computation, uploading of non-private data to the server, and aggregation of this data to produce a new global model. The seminal FL algorithm, *Federated Averaging* (FedAvg) [2], is a client-server algorithm of which many subsequent algorithms can be considered variants.

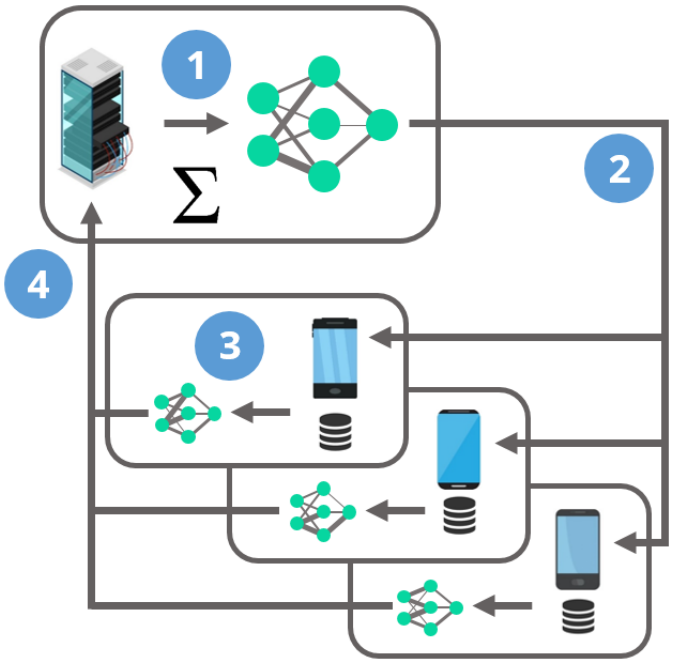

Fig. 1. Operation of FedAvg in the edge-computing scenario. **(1)** The server initialises a single global model. **(2)** The server sends the global model to participating clients. **(3)** Clients perform SGD on their local heterogeneous datasets to produce new models. **(4)** Clients upload their models to the server. The server averages the client models to produce the next round's model.

FedAvg is depicted in Fig. 1. The global model is initialised on the server (Step 1), and downloaded by clients at the network edge over wireless connections (Step 2). Clients perform several steps of Stochastic Gradient Descent (SGD) on their local datasets (Step 3) and upload their new models to the server (Step 4), which averages the received models to produce the new global model.

The per-round convergence rate of the global model can be improved by performing more steps of SGD (K) on clients, resulting in lower total communicated data and training time. The authors of [2] show that the communication cost to reach a target model accuracy can be reduced by up to 95× on some tasks with larger K. However, when data is non-IID across clients, larger K gives diminishing returns for the convergence rate and harms final model performance [2], [3].

We trained a convolutional DNN using FedAvg on a federated version of the CIFAR100 dataset, with 500 non-IID clients [6]. As shown in Fig. 2, increasing K from 1 to 30 resulted in an increased initial convergence rate. However, higher values of K showed worse model performance in the later stages of training, despite faster initial convergence. The primary reason behind this impaired performance is the non-IID data partition on clients, which leads to model divergence during local training. This behaviour is termed *client-drift* [7] and is discussed in the next section. Client-drift must be tackled to better leverage increased local computation on clients and improve the communication-efficiency of FL.

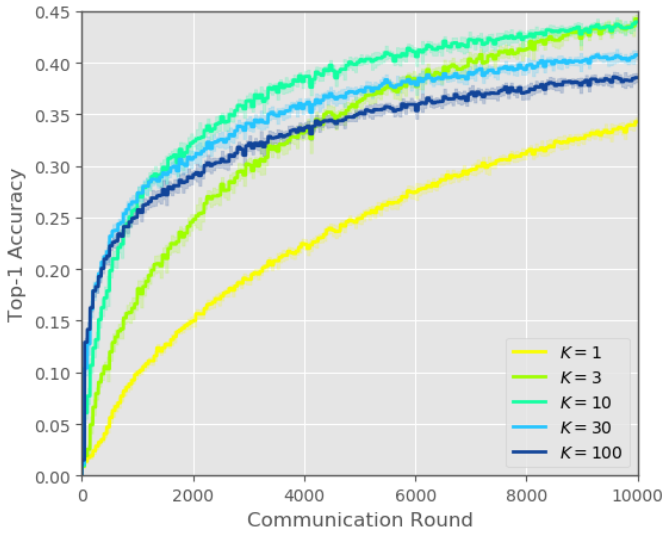

Fig. 2. The effect of increasing the number of local SGD steps (K) performed during FedAvg on the CIFAR100 dataset. Increasing K improves convergence speed at the cost of lower final model performance, with diminishing returns for larger K. Curves show mean values over 10 random trials, with shaded regions giving 95% confidence intervals.

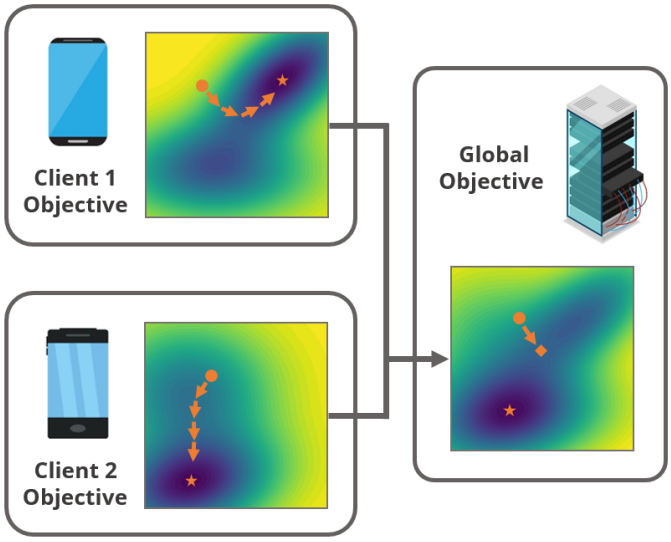

Fig. 3. Demonstration of Client-Drift with two non-IID clients. Each client has a local loss function (contour plots), with the global loss being the average of client losses. The global model at the start of the round is represented by the circle. During local training, the clients' models move towards their minima (stars). The next round's global model (diamond) is the average of client models. Note that the average of client minimisers is not the same as the minimiser of average client losses (star on global loss surface).

B. Client-Drift

When client data is non-IID, the loss surfaces of the ML objectives on clients have different shapes and minimum points. Fig. 3 shows an example 2-dimensional loss surface for two clients. Lower loss is shown as dark blue, with orange stars representing the minimum points of the clients' surfaces.

The global FL objective is defined as the expected loss over all samples on all clients, analogous to the objective normally used in centralised training (minimising the expected loss over all training samples). The surface on the right of Fig. 3 shows the average (expected loss) of the two client surfaces, with the global minimum given by the orange star. Note that none of the loss surfaces are convex.

The orange circles in Fig. 3 represent the starting point of a round of FedAvg. This point is far from each client's local minimum and from the global minimum. During local training the client models move towards their minimisers (orange arrows on client surfaces), which are not identical due to non-IID data. The next round's global model is the average of the two client models (orange diamond). Due to client-drift, the average model is not the minimiser of the global objective.

This simple 2-dimensional example serves to demonstrate the challenge posed by client-drift. For extremely high-dimensional DNNs, where clients perform a few steps of local SGD, client-drift harms the global model convergence rate and the best performance it can reach. Previous works [6]–[8] have derived bounds on the extent of client-drift, showing that it is a function of client data heterogeneity, local learning rate, and the number of local steps. The client-side optimisation strategies discussed later limit the amount of client-drift and hence benefit global model convergence. By improving convergence speed, the number of rounds required for the global model to reach a target accuracy are reduced, and the FL algorithm is made more communication-efficient.

C. Client-Side Optimisation vs. Other FL Algorithms

There are several directions that can be taken to improve FL when considering the scenario and optimisation task (we consider client-side optimisation strategies for their potential to improve the convergence and communication cost of FL).

Some other works propose more sophisticated methods for model aggregation, such as server-based adaptive optimisation, client-model neuron alignment, and model clustering [5]. Alternatively, other algorithms attempt to accelerate the training process by considering the hardware and communications resources of clients. Clients can then be intelligently sampled to minimise the communication and computation time [9]. These model aggregation and intelligent sampling algorithms are executed on the FL server and do not necessarily alter the client-side optimisation process, so are not considered here.

Algorithms for addressing related problems such as Multi-Task FL and Personalised FL have also been suggested [1]. The purpose of these algorithms is not to create a single FL model but multiple models tailored to the non-IID client datasets. We consider these algorithms to be addressing a fundamentally different problem to the original FL task, so are also beyond the scope of this paper.

Several works also propose reducing the per-round model communication cost between clients and server to lower the total communication burden. These can be at the application layer (such as weight sparsification and quantisation) [1], or at the physical layer (such as over-the-air computation to reduce the number of wireless channels used by clients) [4].

Many of the algorithms mentioned above (such as server optimisation, intelligent client sampling, and communication compression) can be used alongside client-side optimisation strategies; an interesting avenue for further research mentioned later in Section V.

TABLE I
RECENT CLIENT-SIDE OPTIMISATION STRATEGIES FOR FL. COMMUNICATION AND MEMORY COSTS ARE GIVEN IN TERMS OF MODEL SIZE ($\boldsymbol{x}$) AND OPTIMISER SIZE ($\boldsymbol{s}$). $\boldsymbol{s}$ DEPENDS ON THE OPTIMISER USED: FOR MOMENTUM-SGD AND RMSPROP OPTIMISERS, $\boldsymbol{s} = x$, FOR ADAM, $\boldsymbol{s} = 2\boldsymbol{x}$.

Algorithm	Approach	Download	Upload	Memory
FedAvg [2]	Baseline	$\boldsymbol{x}$	$\boldsymbol{x}$	$\boldsymbol{x}$
FedProx [10]	Proximal	$\boldsymbol{x}$	$\boldsymbol{x}$	$2\boldsymbol{x}$
FedSplit [11]	Proximal	$\boldsymbol{x}$	$\boldsymbol{x}$	$2\boldsymbol{x}$
SCAFFOLD [7]	Control-Variates	$2\boldsymbol{x}$	$2\boldsymbol{x}$	$3\boldsymbol{x}$
FEDL [12]	Control-Variates	$2\boldsymbol{x}$	$2\boldsymbol{x}$	$3\boldsymbol{x}$
MFL [13]	Local Optimisers	$\boldsymbol{x}+\boldsymbol{s}$	$\boldsymbol{x}+\boldsymbol{s}$	$\boldsymbol{x}+\boldsymbol{s}$
STEM [14]	Local Optimisers	$2\boldsymbol{x}$	$2\boldsymbol{x}$	$2\boldsymbol{x}$
Mimelite [8]	Global Optimiser	$\boldsymbol{x}+\boldsymbol{s}$	$2\boldsymbol{x}$	$2\boldsymbol{x}+\boldsymbol{s}$
FedGBO [15]	Global Optimiser	$\boldsymbol{x}+\boldsymbol{s}$	$\boldsymbol{x}$	$\boldsymbol{x}+\boldsymbol{s}$

III. CLIENT-SIDE OPTIMISATION STRATEGIES

Multiple approaches can be taken to address client-drift and accelerate training within the client loop. We group recent developments into the following broad categories: proximal-based; control-variate; adaptive optimisers local to each client; and global fixed optimisers. These techniques are discussed below, and are summarised in Table I.

A. Proximal-Based

As shown in Fig. 3, client models move towards their respective local minimisers during local training. To reduce client-drift, a proximal term can be added to the client loss function. This term encourages local models to be close to a given point, similar to how L_2 regularisation encourages a model's weights to be of small magnitude.

The first proposed algorithm using proximal updates was FedProx [10], which adds a term to the client loss penalising the squared distance from the local model to the global model, multiplied by a penalty parameter μ. Therefore, the local loss increases when client models move too far from the current global model. The authors of FedProx proved the intuition that μ should be increased as client data becomes more non-IID, to ensure global model convergence on convex and nonconvex objectives. The authors of FedSplit [11] extended FedProx by splitting the local proximal update into a two-step produce, and proved that the minimum points for client objectives are the same as the global minimum point for convex functions optimised via FedSplit.

A major benefit of these proximal methods is that they do not increase the per-round communication cost relative to FedAvg, potentially making them more communication-efficient compared to other methods. However, proximal strategies have shown mixed results for improving model convergence in previous works, which we also find in Section III.

B. Control-Variates

In centralised training, Stochastic Variance-Reduced Gradients (SVRG) reduce the variance introduced by sampling stochastic gradients of the objective function. The global model update in FL has two sources of variance: sampling a subset of all clients per round (especially considering their non-IID data), and the stochastic gradients computed during client training. Therefore, the updates to the global model can have very large variance.

SCAFFOLD [7] was proposed to mitigate the impact of client-drift and stochastic local gradients by keeping a running estimate of the direction of change in the global model. This 'global control-variate' is sent to clients each round alongside the global model. Clients also keep a 'local control-variate', and the difference between the variates gives a measure of client drift. This difference is applied during the client update. SCAFFOLD has provably faster convergence than FedAvg on nonconvex problems, and the authors demonstrated substantial speedup on the popular FL benchmark EMNIST. The FEDL algorithm [12] modifies the client objective with the average global gradient and the local loss using the previous global model. The authors of [12] analytically modelled the convergence of FEDL in a time-sharing wireless-edge environment, incorporating factors such as bandwidth, transmission power, and edge-device CPU frequency to jointly minimise device energy consumption and total training time. The authors then decomposed the resulting optimisation problem to determine the behaviour of devices in different scenarios.

Use of control-variates typically requires FL clients to be 'stateful': local variates need to be stored between communications rounds. Statefulness is a problem in the FL scenario: control-variates are the same size as the model (which can be a very large DNN), and it is assumed that clients would prefer not to have device storage taken up between communication rounds. Transmitting the control-variates between server and clients also increases the per-round communication cost, as shown in Table I.

C. Local Adaptive Optimisers

Adaptive optimisation techniques such as momentum-SGD, RMSProp and Adam are extremely popular for accelerating DNN training, and reduce the need to tune hyperparameters for a given task (adaptive optimisers make DNN loss surface more easily optimised, meaning that their default hyperparameters such as learning rates usually show good performance on many tasks and models).

Adaptive optimisation is straightforwardly extended to FL by applying it during the client update (instead of just using

vanilla SGD as in FedAvg), and averaging the optimiser parameters each round alongside the client models. The authors of MFL [13] show empirically that local momentum can achieve considerable speedup over FedAvg. When the momentum decay parameter of MFL is large, optimiser parameters do not change substantially between aggregations, reducing the impact of client-drift. MFL can easily be extended to other optimisers like Adam. STEM [14] uses client-side momentum, and adds a momentum-step during the server update. The authors of STEM also provided a theoretical analysis of the trade-offs between the number of local iterations and batch size for STEM's communication complexity.

While local optimiser strategies have great potential to accelerate FL, they naturally increase the per-round communication overhead, thus presenting an interesting trade-off between convergence rate and total communicated data. These methods are compatible with a variety of optimisers, and as shown in Table I, optimisers have different per-round communication costs. This adds another communication-cost *vs.* convergence decision for FL engineers.

D. Global Adaptive Optimisers

With local adaptive optimisers, optimiser parameters are updated within the client training loop and are averaged alongside the model parameters. If the decay hyperparameters are large, optimiser values will not change much during the local update. However, if there are a large number of local updates or the decay hyperparameter is small, local optimisers could potentially contribute to client-drift. To address this problem, optimiser parameters can be kept constant during the local training loop and updated between communication rounds.

Mime [8] uses a fixed (global) optimiser in this way. Clients compute an unbiased full-batch gradient on their local data each round before performing local updates with a fixed optimiser. They then upload these full-batch gradients and models to the server. The server uses the full-batch gradients to update the global optimiser each round. The authors of Mime propose two variants: Mimelite and Mime (which also incorporates SVRG). Both increase the communication and computation costs of FL due to computing full-batch gradients and transmitting them to the server. We proposed FedGBO [15], which reconstructs the average client gradients on the server for updating the global optimiser. FedGBO's optimiser is biased as it incorporates the biased local gradients, but FedGBO demonstrates rapid convergence on various benchmark FL tasks, and reduces the upload and compute costs compared to Mime. Reducing the upload cost in FL is important due to lower upload bandwidth at the wireless edge (compared to the download bandwidth).

As with local optimiser methods, different optimisers also change the per-round communication cost of global optimisers because of the need to transmit the global optimiser values.

IV. Experimental Evaluation

In this section, we provide an experimental comparison between some of the client-side optimisation strategies described in Section III on two popular FL benchmark datasets.

A. Setup

We simulate FedAvg and one algorithm from each of the categories in Section III: FedProx [10] (proximal), SCAFFOLD [7] (control-variate), MFL [13] (local optimisers) and FedGBO [15] (global optimiser), on the CIFAR100 and Shakespeare benchmark FL datasets using GPU-equipped workstations.

CIFAR100 is an image-classification task consisting of (32×32) pixel images of objects from 100 classes, split into 500 clients according to the class labels, using the procedure from [6]. We train a Convolutional DNN with $K = 10$ local steps. Shakespeare is a next-character prediction task using the plays of Shakespeare. The text is partitioned into 660 clients according to the speaking parts in each play, with a sequence length of 80, using the procedure from [3]. We train a Gated Recurrent Unit (GRU) DNN with $K = 100$. A larger K is required due to the much greater number of samples in the dataset.

We tuned the algorithms' hyperparameters to reach the maximum validation accuracy within 10k communication rounds. Each hyperparameter was tuned in the range $(10.0, 3.0, \cdots, 0.003, 0.001)$, except for β of MFL and FedGBO (using the momentum-SGD optimiser), which was tuned in the range $(0.6, 0.9, \cdots, 0.996, 0.999)$. The specific values used for the CIFAR100/Shakespeare experiments were: FedAvg (learning rate $\eta = 0.03/0.03$); FedProx (learning rate $\eta = 0.03/0.03$, proximal term $\mu = 0.01/0.001$); SCAFFOLD (client learning rate $\eta = 0.003/0.03$, server learning rate $\eta_g = 3.0/1.0$); MFL (learning rate $\eta = 0.03/0.03$, momentum decay term $\beta = 0.99/0.996$); FedGBO (learning rate $\eta = 0.03/0.1$, momentum decay term $\beta = 0.99/0.99$).

B. Convergence Results

As shown in Fig. 4 (a), client-side momentum-SGD (SGDm) - as used in MFL (grey) and FedGBO (yellow) - increased the global model's convergence rate and the maximum accuracy achieved in 10k communication rounds for CIFAR100. FedGBO reached 52% accuracy, and MFL 50%, compared to FedAvg's (red) baseline of 45%. SCAFFOLD (purple) showed slower initial convergence than FedAvg, but superior final accuracy. FedProx (blue) showed no benefit over FedAvg, and when tuning μ the best performance was usually seen with very small μ, making FedProx almost equivalent to FedAvg.

The convergence curves for the Shakespeare dataset in Fig. 4 (b) show less dramatic differences between algorithms. Neither MFL nor FedProx were able to improve convergence compared to FedAvg. As there were $K = 100$ local SGD steps performed in Fig. 2 (b), there is potential for large client-drift. The local optimisers of MFL likely did not help to reduce client-drift and hence performance was poor. On the other hand, it appears that the global optimiser of FedGBO and the control-variates of SCAFFOLD were successful in accounting for client-drift, showing substantial performance benefit.

C. Performance vs. Cost Trade-offs

Table I shows that client-side optimisation strategies come with different per-round communication costs. The total communication cost of an algorithm is the product of the per-round cost and the number of rounds. If an algorithm improves

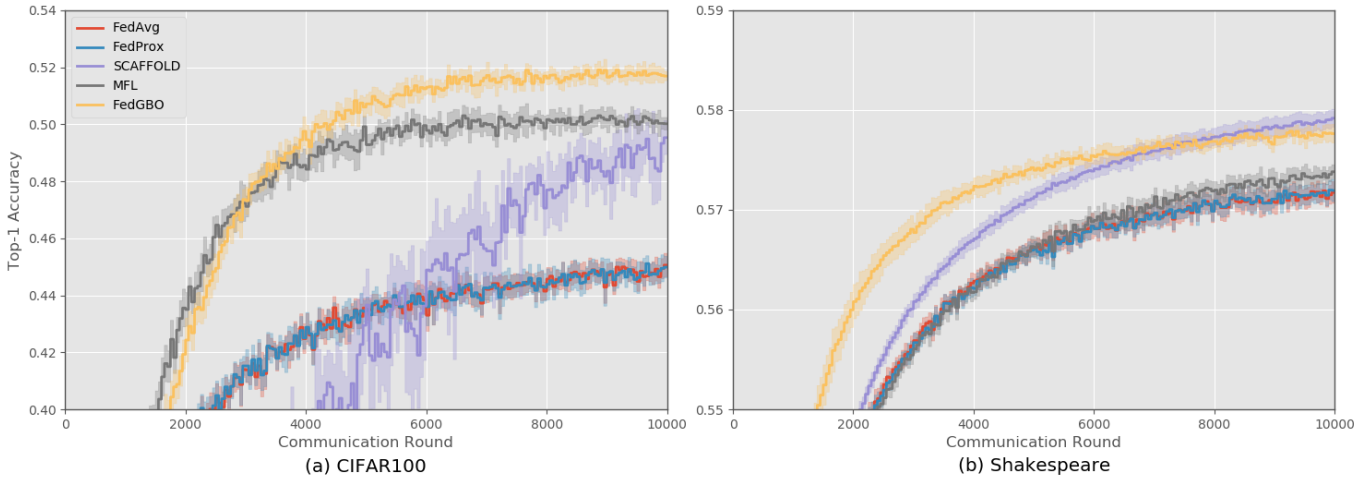

Fig. 4. Convergence of different client-side optimisation strategies on the CIFAR100 and Shakespeare datasets. Curves represent mean over 10 random trials, and shaded regions represent 95% confidence-intervals of the mean.

the convergence rate (hence reducing the number of required rounds) but comes at the cost of higher per-round communication, these factors must be weighed against each other.

FedProx has the same communication cost as FedAvg, but adds another hyperparameter to tune and has a larger memory requirement. Considering FedProx showed no convergence benefit over FedAvg, it is a poor choice for both scenarios.

SCAFFOLD and MFL (with SGDm) both increase the up/downloaded data compared to FedAvg by 2×. For CIFAR100, considering that both were able to achieve significantly higher accuracy than FedAvg, the extra communication cost may well be acceptable. For Shakespeare, SCAFFOLD's more modest improvement represents a trade-off. An interesting comparison is the point at which each algorithm reaches 57.2% accuracy (FedAvg's maximum). FedAvg reached this at 10k rounds, whereas it took SCAFFOLD approximately 5k rounds. Therefore, the total data communicated by each algorithm to reach 57.2% accuracy is roughly equivalent, but SCAFFOLD achieves it in half as many rounds (which could mean significantly less time in a real-world FL setting).

FedGBO (with SGDm) has a 2× download cost compared to FedAvg, but equal upload cost. Considering the asymmetric upload and download bandwidths at the network edge, the higher download cost could be considered less detrimental. FedGBO provided clear improvement for CIFAR100, and taking the 57.2% accuracy mark for Shakespeare, represents a significant reduction in total data and communication rounds compared to FedAvg and SCAFFOLD.

V. Conclusion & Future Work

In this paper, we provided an overview of Federated Learning (FL), which has the potential to train machine leaning models from distributed data whilst preserving client data privacy. We identified the key challenges faced by FL algorithms at the wireless edge, and provided a tutorial on the statistical problem of 'client-drift', which arises from heterogeneous client data and reduces the communication-efficiency of FL. We then surveyed state-of-the-art algorithms that tackle client-drift and improve FL's convergence rate, discussing their benefits and drawbacks in terms of communication and memory requirements. We finished by performing a set of comparison experiments on two benchmark FL datasets, discussing the trade-off between communication cost and convergence rate which is central to FL. This work offers valuable insight into the rapidly developing sub-field of client-side optimisation in FL and its relation to wireless communications, and gives rise to some open challenges and research directions:

- Many of the surveyed client-side strategies improve the convergence of FL at the cost of increased per-round communication. Compression techniques for FL have been extensively studied [5], but there are very few works investigating compression alongside novel client-side optimisation. Papers considering physical-layer communication-reduction (e.g. Over-The-Air computation) also largely do not consider novel client-side optimisation. Furthermore, combinations of different client-side techniques (along with server-side techniques) could be explored for even better performance.
- The majority of FL benchmark datasets take existing datasets and devise a non-IID splitting procedure for clients. However, data generated by wireless edge devices (such as remote-sensors) have their own unique properties and distribution, and to the best of our knowledge no benchmarks like this yet exist. Realistic datasets for this domain would allow more relevant comparison of FL algorithms at the wireless edge. Additionally, most previous works provide simulated results for algorithms, but realistic testing in real-world deployments or with wireless testbeds under various conditions would shed light on the true performance of these algorithms and provide new directions for improvement.
- Novel client-side optimisation strategies could be developed that better leverage the network structure of the edge. For example, optimisation strategies that make use of higher bandwidth connections to nodes that are closer

together on the network, and the asymmetric transmission speeds of edge nodes. The theoretical convergence properties of these algorithms given their communication environment may be of particular interest.

ACKNOWLEDGMENTS

This work was supported by EPSRC DTP Studentship and the EU Horizon 2020 INITIATE project under the Grant Agreement No. 101008297. The European Union Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[2] B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. International Conference on Artifical Intelligence and Statistics (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[3] S. Caldas, P. Wu, T. Li, J. Konecný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," in *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

[4] Z. Qin, G. Y. Li, and H. Ye, "Federated learning and wireless communications," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 134–140, 2021.

[5] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1342–1397, 2021.

[6] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[7] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. International Conference on Machine Learning (ICML)*, vol. 119, 2020, pp. 5132–5143.

[8] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," *arXiv e-prints arXiv:2008.03606*, 2020.

[9] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2021.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Machine Learning and Systems (MLSys)*, vol. 2, pp. 429–450, 2020.

[11] R. Pathak and M. J. Wainwright, "Fedsplit: an algorithmic framework for fast federated optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7057–7066, 2020.

[12] C. T. Dinh, N. H. Tran, M. N. H. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2021.

[13] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating Federated Learning via Momentum Gradient Descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, aug 2020.

[14] P. Khanduri, P. Sharma, H. Yang, M. Hong, J. Liu, K. Rajawat, and P. K. Varshney, "Achieving optimal sample and communication complexities for non-iid federated learning," in *ICML Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2021.

[15] J. Mills, J. Hu, G. Min, R. Jin, S. Zheng, and J. Wang, "Accelerating federated learning with a global biased optimiser," *arXiv e-prints arXiv:2108.09134*, 2021.

Jed Mills is a Computer Science Ph.D. student in the Department of Computer Science at the University of Exeter, UK. He received a B.Sc. in Natural Science from the University of Exeter in 2018. His research interests include machine learning, federated learning and mobile edge computing.

Jia Hu is a Senior Lecturer in Computer Science at the University of Exeter. He received his Ph.D. degree in Computer Science from the University of Bradford, UK, in 2010, and M.Eng. and B.Eng. degrees in Electronic Engineering from Huazhong University of Science and Technology, China, in 2006 and 2004, respectively. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science at the University of Exeter, United Kingdom. He received his Ph.D. degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.