BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 02/10 pp404–420
Volume 6, Number 4, December 2023
DOI: 10.26599/BDMA.2022.9020046

Elastic Optimization for Stragglers in Edge Federated Learning

Khadija Sultana*, Khandakar Ahmed, Bruce Gu, and Hua Wang

Abstract: To fully exploit enormous data generated by intelligent devices in edge computing, edge federated learning (EFL) is envisioned as a promising solution. The distributed collaborative training in EFL deals with delay and privacy issues compared to traditional centralized model training. However, the existence of straggling devices, responding slow to servers, degrades model performance. We consider data heterogeneity from two aspects: high dimensional data generated at edge devices where the number of features is greater than that of observations and the heterogeneity caused by partial device participation. With large number of features, computation overhead on the devices increases, causing edge devices to become stragglers. And incorporation of partial training results causes gradients to be diverged which further exaggerates when more training is performed to reach local optima. In this paper, we introduce elastic optimization methods for stragglers due to data heterogeneity in edge federated learning. Specifically, we define the problem of stragglers in EFL. Then, we formulate an optimization problem to be solved at edge devices. We customize a benchmark algorithm, FedAvg, to obtain a new elastic optimization algorithm (FedEN) which is applied in local training of edge devices. FedEN mitigates stragglers by having a balance between lasso and ridge penalization thereby generating sparse model updates and enforcing parameters as close as to local optima. We have evaluated the proposed model on MNIST and CIFAR-10 datasets. Simulated experiments demonstrate that our approach improves run time training performance by achieving average accuracy with less communication rounds. The results confirm the improved performance of our approach over benchmark algorithms.

Key words: edge computing; federated learning; distributed machine learning; regularization; stragglers

1 Introduction

Recent years have witnessed applications of machine learning (ML) algorithms to the unprecedented data generated at edge devices. The wide application areas include image classification[1,2], natural language processing[3,4], autonomous driving[5,6], and human-computer interaction[7,8]. ML models intended for such areas need massive amount of data which need to be available at training time[9]. As per general data protection regulation (GDPR) rules, the data have to be protected[10–12]. Therefore, federated learning (FL)[13] emerged as a solution to data privacy in ML training. As can be seen in Fig. 1, federated learning has caught attention and its interest is increasing worldwide due to its privacy-preserving characteristics[14–16]. Although FL can solve privacy issues, the architecture adopted is highly rigid. FL has been widely adopted in various domains in collaboration with blockchain such as cognitive computing[17] and fog

• Khadija Sultana and Hua Wang are with the Institute for Sustainable Industries and Liveable Cities (ISILC), Victoria University, Melbourne 3011, Australia. E-mail: khadija.sultana@live.vu.edu.au; hua.wang@vu.edu.au.
• Khandakar Ahmed is with the Intelligent Technology Innovation Lab (ITIL), Institute for Sustainable Industries and Liveable Cities (ISILC), Victoria University, Melbourne 3011, Australia. E-mail: khandakar.ahmed@vu.edu.au.
• Bruce Gu is with Shandong Computer Science Center (National Supercomputer Center), Jinan 250101, China. E-mail: gusj@sdas.org.
* To whom correspondence should be addressed.
Manuscript received: 2022-06-30; revised: 2022-10-17; accepted: 2022-11-11

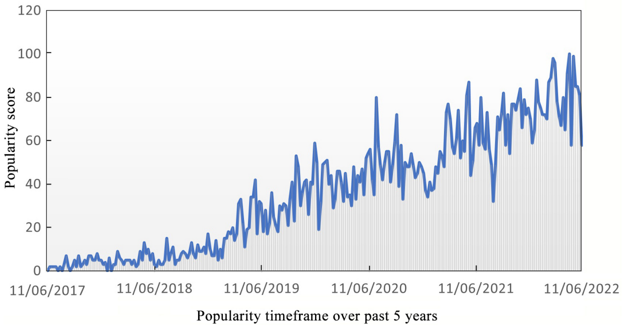


Fig. 1 Federated learning worldwide.

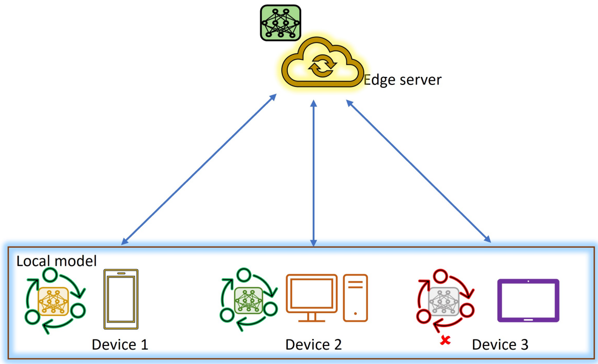


Fig. 2 Stragglers in EFL.

computing[18]. Traditionally, FL follows a parameter-server architecture where a cloud server is responsible for model aggregation. However, the cloud associated communication latency is quite high[19,20]. Originally, FL is designed for wired connected systems. Recently, FL has been applied as an application in edge computing connecting to wireless links. This overcomes traditional architectural problems and also provides intermediate edge level aggregations improving communication speed. Therefore, a hot research area called edge federated learning (EFL)[21] which implements federated learning in edge computing is widely adopted. Most of the recent research in FL has focused on EFL framework[22–28]. In addition to this, authors in Refs. [29, 30] described current privacy preserving research in edge computing using federated learning.

In EFL, edge servers are responsible for intermediate model aggregations. Edge devices communicate with closer edge servers rather than cloud servers thereby decreasing communication latency. The problem in EFL is the efficient utilization[31] of resources to improve performance. Furthermore, performance is degraded due to system and statistical heterogeneity. Ideally, in EFL each client update is required for global model update. Currently, a benchmark algorithm for FL, federated average (FedAvg), requires all the participating edge devices to perform same number of training rounds regardless of their system and statistical heterogeneity. The mentioned heterogeneities stem stragglers who cannot complete their all training rounds thereby delaying global model aggregations. Stragglers have been a topic of interest in distributed systems from past few years. The interest of stragglers in FL grew quite fast in recent literature. We are interested in edge federated learning (EFL) where FL is an application in edge computing. Figure 2 represents the straggling device scenario where Device 3 takes more time delaying global model aggregation. The research work for preserving privacy has been a hot topic for many years[32–34], it has also been discussed in edge framework[35]. In EFL, stragglers issue can intensify degradation of model performance as the participating devices are in millions and uncertain about their prolonged participation as well as their complete time dedication for model training. System heterogeneity arises from different computational capabilities of clients—having different system configurations. Similarly, statistical heterogeneity is due to the high dimensional data with large number of features or parameters. For instance, images are stacked on top of each other forming video where pixel values stored could be very complex. Hence, the processing time for such kind of data also requires devices to completely dedicate their system resources. Any limitation in system resources or complexity in data can manifest stragglers thereby decreasing model utility. Large number of parameters from high dimensional data have been a statistical problem in literature, where the use of efficient methods to identify appropriate required features was adopted. Many deep neural networks used for federated model training have large number of parameters, millions in number due to high data dimensionality[36–38]. These large number of parameters make it challenging for model training. Deep learning is often helpful in dealing with feature extraction so that a model with optimal features is selected with feasible number of parameters[39]. In Ref. [40], the authors talked about client-drift due to high-dimensional data together with few local steps hurts global model convergence. Further, high data dimensionality of model parameters causes communication latency in mobile edge computing between base stations and edge devices[41]. In Ref. [42], clients are grouped as per their data distributions and then their individual model is obtained from each group. The benefit of doing this clustered approach for model aggregation is that the high dimensional model parameters from clients which belong to other group cannot reach other distribution group thereby improving the performance of model training.

Although an EFL is envisioned as a promising solution for faster and scalable model aggregation, it suffers from the straggling-effect. That is, devices with heterogeneity in data and system can cause delay communicating their local updates to edge servers. Stragglers can cause a major hindrance in EFL implementation by causing the slow-down of run time performance. Many methods have been considered for the above-mentioned challenges such as synchronous updates, asynchronous communication, and hybrid approaches (combining synchronous and asynchronous behaviors) to combat stragglers. Recent literature dealt with the straggler issue by exploiting the edge computing benefits. For instance, in Ref. [43], the FL made straggler efficient by offloading the training process at edge servers. In Ref. [44], the straggling-effect associated with communication heterogeneity is studied and re-configurable intelligent surface (RIS) is employed for efficient model aggregation and performance improvement. Similarly, for over-the-air FL, in Ref. [45] straggling edge devices are assisted with multiple relays for uploading model parameters to a server. Therefore, communication heterogeneity in EFL is dealt in Ref. [45]. In Ref. [46], the stale gradients of stragglers are used to improve model utility. However, gradient staleness can further increase model complexity and decrease run-time performance due to difficulty in finding the global optima. Although, the existing work focused on various techniques for mitigating stragglers such as asynchronous updates[47], ridge shrinkage[48], sparsity for efficiency[49], incorporating fasters clients according to their speed[50], and ridge penalization[48] for smoothness of objective function thereby improving its convergence properties. In addition to this, to avoid any bottleneck caused by centralized FL, blockchain enabled asynchronous federated learning was introduced in Ref. [51].

From regularization perspective, some studies used lasso (L1 regularization) and others used ridge (L2 regularization)[48]. In other words, when the number of predictors exceed the number of observations, the lasso does not perform well. Further, nor ridge regression can work alone when parameters are larger in data. Therefore, individually, lasso and ridge do not perform well. In literature, focus is on usage of lasso or ridge to obtain efficiency. However, the advantages of both combined methods have not been considered. Therefore, elastic net is the combination of both lasso and ridge penalization. The application of elastic net in EFL is remained unexplored. In this paper, we focus on application of elastic net regularization in EFL which consists of combination of lasso and ridge penalty in local objectives of edge devices. Both lasso and ridge are effective for statistical learning in literature. Elastic net is a successful method in sparse representation and works well when the number of parameters is large. So, elastic net regularization works well by balancing between lasso and ridge. The key contributions of this paper are as follows.

- We identify and formulate stragglers as a distributed optimization problem in edge federated learning. Stragglers are edge devices whose response to an edge sever for model aggregation is slow. The delay in response is due to data heterogeneity associating large number of parameters and gradient divergence due to partial submissions
- We propose elastic optimized edge federated learning (FedEN) approach to deal with stragglers due to data heterogeneity contributed by today's high dimensional data and partial edge device participation. FedEN mitigates stragglers by dealing with large number of features with the help of lasso penalization generating sparse efficient models. In addition to this, edge devices stay near to local optima with the help of ridge penalization thereby restricting the strict feature elimination of lasso. Hence, elastic optimization in EFL gives an optimal balance between lasso and ridge thereby benefiting EFL with elimination of stragglers.
- Our extensive experiments for image classification using FedEN prove that our work gives substantially higher accuracy and lower loss compared to benchmark algorithms.

With FedEN, improved training performance with sparse models updates communicated and grouping correlated variables. We conduct experiments on MNIST and CIFAR-10 datasets using convolutional neural network (CNN) with stochastic gradient descent (SGD) as an optimizer. Results demonstrate that FedEN has improved run time performance compared to baselines. The rest of the paper is organized as follows. Section 2 describes the straggler mitigation in literature. Section 3 introduces the problem of stragglers in edge federated learning and the elastic federated learning algorithm for the proposed framework. Section 4 demonstrates the experimental results and the conclusion is in Section 5.

2 Related Work

In this section, we go through the research studies dealing with stragglers in literature. There has been innumerable methods used for mitigation of straggles. Some methods are efficient to deal with stragglers while others simply ignore them from research studies.

The problem of stragglers has been studied by many research works in literature as a distributed optimization problem. Reference [52] proposed the concept of back up worker to mitigate stragglers. The main concept was to increase training speed by introducing redundancy in terms of additional participating clients. In this approach, a total of $(N + b)$ clients are considered where b indicates extra clients for redundancy. During training process, whenever first N clients respond back, the server stops waiting for the other updates and performs global aggregation while discarding the updates received later. Whereas in Ref. [53] exploitation of stragglers is focused by fixing the computation and communication deadline so that all workers complete training in same duration of time and communicate with server without any wait time. This reduces wastage of resources as stated in earlier approaches. Additionally, the former approach works well only when stragglers are less than or equal to the back up workers, b, considered. If there are more than b then this resembles to an ideal stragglers situation where the wait time is introduced. Moreover, both of them follow a synchronous distributed training approach and are approaches followed in distributed systems. However, these are also applicable to the context of FL where distributed training is performed without data sharing. Some of the studies such as Ref. [54], tried avoiding the straggler problem. In another study, the approach of adjusting the training batch was studied according to computation time taken by clients in Ref. [55] where the identification of both static and dynamic stragglers have been performed. However, the adjustment depends on the computation time of the workers at the previous iteration which takes more time to decide the mini batch size and update it in subsequent rounds. While in Ref. [56], the training speed of each worker is managed by a parallelism manager. If stragglers are detected by comparing the remaining time of a worker with the standard epoch time, its task is transferred to the fast clients to speed up the training process. This is similar to Ref. [57], where the data duplication among clients is seen as a way to mitigate straggling problem in distributed systems. The algorithms for straggler mitigation in distributed system mainly involve data duplication among clients, tasks re-assignment, and ignoring or dropping stragglers (as in case of federated averaging). These methods are not applicable in the context of FL since data are not considered as a central entity at cloud server. Hence, the problem of stragglers in FL is unique with its own characteristics and can be deteriorating in case of federated edge computing context. Since FL in edge computing (EC) can cause the various issues such as computation and communication delays due to straggling nodes at the edge levels[58], the intermediate edge level communications remain halted thereby delaying the cloud communication at the next level. In Ref. [59], in order to deal with stragglers in EC scenario, the number of clients selected by the edge serves is considered to be large so that even if the straggling nodes happen to arise, it shall not have major impact.

To alleviate stragglers in FL, which could be thought of as a distributed optimization without data centralization and dissemination among clients, various techniques have been implemented. For instance, ESync[60] which involves dealing with blocking time of pioneers instead of focusing on the long-tail (straggling) effect was recently proposed. ESync tries to utilize the idle time of pioneers by allowing them to perform additional training rounds only if it can be accommodated within the predicted straggler response time.

Based on the straggler response time, the state server decides if the training round of the pioneer could be proceeded or allowed to synchronize. It is worth mentioning that if pioneers are allowed to perform too much local training during the wait time for stragglers, it could diverge from the local minimum thereby adding inaccurate results to global model. A different approach from ESync is introduced in TiFL[50]. TiFL selects the clients with similar response time into a same tier such that whenever the clients are randomly selected from a tier, they would take same time for returning the updates to the server. Unlike all of the above methods, FedProx[48] introduces a proximal term to deal with system and statistical heterogeneity which are the causes towards straggling-effect. The notion behind addition of proximal term to FedAvg is reparameterizing it such that the local updates are not diverged from the global objective, thereby involving partial work of the clients by tuning it.

While others focus on profiling clients into tiers (TiFL), restricting clients to diverge from global objective (FedProx), and utilizing block time for pioneers (ESync), FedCS[47] focuses client selection as an important step for the federated training in mobile edge computing. The selection ensures that only clients who can perform model training and update within the set deadline are selected. In this way, more clients can be incorporated in the training process to achieve high performance results unlike the baseline FL protocol with random selection restricting the number of clients per round. Some of the recent literature that dealt with stragglers in edge networks are Refs. [61–63].

There are various studies in literature as mentioned before[26, 27, 31, 45, 46] that have dealt with stragglers in edge federated scenario. Inspired by the above mentioned studies, we propose elastic net optimization algorithm for federated learning (FedEN). FedEN is a customized version of FedAvg[13] and closely related to FedProx[48] consisting of tuning parameter λ and mixing parameter α. Compared to FedProx, FedEN tries to optimize the local objective such that the sparse models are obtained and at the same time parameters are shrinked to reach as close as possible to the local optima. Algorithm 1 shows the general traditional federated averaging technique and Table 1 shows the notations used in the paper.

Algorithm 1 Federated averaging

Input: B, N, E, and η

Output: federated trained global model

1: Target accuracy not achieved
2: Global model parameters broadcast to edge devices
3: Local model training at each client
4: **if** complete local training after E updates **then**
5: send the updated parameters to the edge server
6: **if** local updates received by the edge server **then**
7: perform aggregation using Eq. (3)
8: **end if**
9: **end if**

Table 1 Notations.

Notation	Description
λ	Tuning parameter
β'	Lasso solution
τ	Subgradient from KKT optimality solution
A	Predictor matrix
β'^{LARS}	LARS lasso solution
ϵ	Optimal stability point
η	Step size
s_g	Stochastic gradient
N	Total edge devices
T	Total communication rounds
L	Loss function
X	Algorithm
$D = d_1, d_2, \ldots, d_n$	Data samples
n	Observations
p	Features
$F(w)$	Objective function
$F'(w)$	Derivative of objective function (gradient)
M_{total}	Centralized data model
M_{global}	Federated trained model
Y	Output predictor
LSS	Least square estimate
RSS	Residual sum of squares
EFL	Edge federated learning

3 System Modeling and Analysis

In this section, we describe the general edge federated learning framework and stragglers in the proposed framework and introduce the straggler-resistant optimization algorithm. We first describe the learning based edge computing and federated learning to propose edge federated learning in the subsequent sections.

In edge computing[64], for training a machine learning (ML) model[65], data are offloaded at the edge server. The edge server then trains the offloaded data from the edge devices. The benefit from this type of offloading is that the data are easily accessible rather than being downloaded from a centralized server. Further, the communication time with the centralized server is greatly reduced with the introduction of server at the edge level. The bandwidth is improved and latency is decreased. In federated learning[13], the clients utilize their local datasets and perform local training instead of transferring the data to the server for training. The main advantage is the privacy of the local data as it is not transmitted anywhere for training.

3.1 Edge federated learning

From the perspective of performance, the above two schemes are limited by either computation or communication barrier. In edge computing learning, offloading data to the edge server can cause the privacy issues. Further, the data uploading time is relatively high if the datasets are large enough. On the other hand, the parameter server architecture of FL is rigid and creates a bottleneck while communicating the training results and downloading the initial model parameters.

We consider an edge federated learning (EFL) context which consists of edge server and N edge devices available for the federated learning (FL). These N edge devices can be represented as $N = \{1, 2, \ldots, N\}$. Only $S \subset N$ are randomly selected for the training process. Let T be the communication rounds required for model aggregation and the index for representation be t. E is the total number of updates performed by the edge devices for local training with the index e. The model parameter is represented as ω. For an edge device i in round t with local update step e, the model parameter is represented as $\omega_{i,e}^t$.

The main objective for efficient EFL is to minimize the loss which can be formulated as follows:

$$\min \frac{1}{N}\sum_{i=1}^{N} f_i(\omega) \tag{1}$$

where $f_i(\omega)$ is the loss function for device i for the sample (x_i, y_i).

For j edge devices, the loss function is

$$F_j(\omega) = \frac{1}{N_j}\sum_{i \in \beta_j} f_i(\omega) \tag{2}$$

The global loss is defined as the summation over all the local loss functions

$$f(\omega) = \sum_{j=1}^{J} \frac{n_j}{n} F_j(\omega) \tag{3}$$

Generally, there are three steps in edge federated learning training. They are (1) broadcast of model parameters by the edge server to the clients, (2) local model training at the client side and sending the updated model to the edge server, and (3) model aggregation by the edge server and repeating Steps (1) to (3) until target accuracy is reached. Figure 3 shows the FL performed at an edge level. The local update performed by the clients can be modelled as follows:

$$\omega_i^{t+1} \leftarrow \omega_i^t - \eta_{\phi_i}(\omega_i^t) \tag{4}$$

where ω_i^{t+1} is the model parameter for the edge device i for rounds $t+1$. After E local model updates as in Eq. (2), the updated parameters are then aggregated by the edge server as follows:

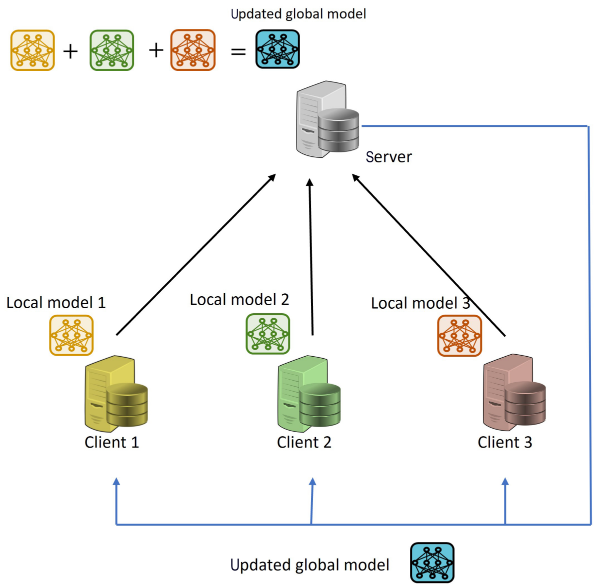


Fig. 3 FL at each edge level.

$$\omega^{t+1} \leftarrow \omega^t + \frac{\eta_\phi}{|S|}\sum_{i \in S}(\omega_i^t - \omega^t) \tag{5}$$

The aggregated model parameters then form a global model whose model parameters are again broadcasted to the edge devices in an iterative manner.

Local SGD: According to Ref. [66], local SGD can be a critical component of federated learning. Further, the addition of regularization term in local SGD can improve generalization[67]. Local SGD applies the stochastic gradient descent as a local optimizer. However, there is no guarantee that if local SGD being used can guarantee convergence without any assumptions. Therefore, the assumption on the gradient bounds and dissimilarity is assumed in most of the works in recent literature. For instance, in Ref. [68], gradient dissimilarity is assumed and the bound is made on the gradients. Generally, the common assumptions of lipschitzness, $\|\nabla F_j(\omega) - \nabla F_j(\omega^{|})| \leqslant L\|\omega - \omega^{|}|$, are assumed for any of the theoretical guarantees, followed by bounded and dissimilar gradients.

Heterogeneity and convergence: As mentioned in earlier sections, we consider system and statistical heterogeneity in our work. Incorporation of system heterogeneity implies that the devices who cannot complete their required number of local iterations in training can perform any random number of iterations and contribute in the model aggregation. And, high dimensional data with larger number of parameters are long time statistical problem in literature which contributes to statistical heterogeneity causing straggling-effect. The divergence of edge devices from the local objective is due to the statistical heterogeneity which worsens when the number of model parameters is very large and computationally-inefficient straggling devices contribute their incomplete results in model aggregation. In literature, different notations are used to specify the intensity or degree of heterogeneity as per the variances in the local objective assumptions. For instance, in Ref. [48], bounded dissimilar gradient assumption was analyzed. Whereas, Ref. [69] first used the term gradient diversity which measures the dissimilarity degree of individual gradients of the loss functions. Therefore, the gradient is larger when the products between the gradients are small. Its ratio is defined as follows:

$$\nabla(\omega) = \frac{\sum_{i=1}^{N} ||\nabla f_i(\omega)||_2^2}{||\sum_{i=1}^{N} \nabla f_i(\omega)||_2^2} \tag{6}$$

3.2 Elastic federated learning

We formally define edge federated learning straggler problem, in which S randomly chosen edge devices collaboratively participate for training a classification model and delay communication with the edge server. The reason for straggler is the computation or statistical heterogeneity. We consider statistical heterogeneity contributors as the devices that perform more explorations while local training and the re-participation of the straggling devices which in turn increases the data correlation. The goal is to minimize the local loss and improve the training performance by achieving the target test accuracy with reduced communication rounds.

The edge devices have statistical variances in their local data which can facilitate straggler manifestation. We allow the local edge devices to perform more exploration to reduce the number of communication rounds with the edge server. Moreover, to limit the effect of statistical variance, elastic penalization is used in the local optimizers for better performance. We consider generalized linear models (GLM) in FL. FedEN is equivalent to introducing penalization via lasso (L1) and ridge (L2) on the model weight w. Let $\tilde{l}(w, x, y)$ be the elastic net local objective loss which is equivalent to the expectation of the original loss consisting of varied parameter weights obtained from the global model. That is

$$\text{Local objective} = \tilde{l}(\omega, x, y) \tag{7}$$

where ω is the model parameter and y is the predictor for an input feature x. Taking expectation, the loss is represented as follows:

$$E_j[l(\omega, x, y)] = l(\omega, x, y) + \pi(\omega) \tag{8}$$

And

$$\pi(\omega) = \lambda\left[(1-\alpha)/2||\omega||^2 + \alpha||\omega||\right] \tag{9}$$

where $l(\cdot)$ can be any suitable loss function and $\pi(w)$ is the elastic net penalty over the loss function consisting of the mixing parameter α in the penalization. In Eq. (9), λ is the tuning parameter with $\lambda \geqslant 0$ and α is the mixing parameter with $0 \leqslant \alpha \leqslant 1$. The function in Eq. (8) with the assumption of strongly convex and choice of mixing parameter becomes $f_j(w, w^t)$, namely the hessian function. So, from Eq. (9), λ and α relate to the strength of the regularization. In general, $\alpha = 0$ means the application of L2 on the model weights. Moreover, when $\alpha = 1$, the sparsity based lasso is imposed on the model weights thereby creating the sparsity in the local models. With $\lambda > 0$ and $\alpha < 1$, the elastic net problem has a unique solution irrespective of the correlations.

EN is particularly useful when the dataset is high dimensional with predictors (p) are larger than the observations (n). As mentioned earlier, it is a combination of lasso (L1) and rigde (L2) penalty. Both L1 and L2 are effective for statistical learning in literature as a good balance between them can achieve sparse models with better prediction accuracy. Elastic net has a tuning parameter λ and a mixing parameter α. The mixing parameter α regulates the amount of weight given to lasso and ridge. With $\alpha = 0$, elastic net is equivalent to ridge regression. Similarly, when $\alpha = 1$, elastic net penalization becomes special case of lasso thereby imposing only L1 penalty on the model parameters while training. L1 penalty helps in better feature selection thereby giving sparse results. However, the L2 penalty adds smoothness to the objective function by shrinking the model parameters. Therefore, ridge penalty keeps the model parameters shrinked in a way that the gradient divergence is reduced and the local optima is achievable.

We employ the following standard assumptions[48, 70–72] for edge federated learning algorithm.

Assumption 1 (lipschitz gradient): The function f is L-smooth if its gradient is L-lipschitz continuous, i.e., for any two model parameters ω and $\omega^{|}$, $F_j(\omega)$ is the lipshitz gradient for each edge devide j such that $j \in$

$1, 2, \ldots, N$. That is, $|\nabla F_j(\omega) - \nabla F_j(\omega^{|})| \leqslant L|\omega - \omega^{|}|$.

Assumption 2 (B—gradient dissimilarity): The local gradients are B-dissimilar from each other. That is, $|\nabla F_j(\omega)| \leqslant B|\nabla f(\omega)|$.

Assumption 3 (β—inexact local solutions): For each device j such that $j \in \{1, 2, \ldots, N\}$ and t communication rounds, local training results in β dissimilar solutions, i.e., $|\nabla f_j(\omega_j^{t+1}, \omega^t)| \leqslant \beta|\nabla f_j(\omega, \omega^t)|$.

Assumption 4 (δ—bounded hessian): For a strongly convex Function Eq. (8) with the smallest eigen-value and the indentity matrix I, the bounded hessian can be defined as $\nabla^2 F_j(w) \succeq -\delta I$.

The loss function in Formula (1) is strictly convex and differentiable. For any prediction matrix A, sub-gradient Γ, lasso solution β', the Karush-Kuhn-Tucker (KKT) conditions for Formula (1) can be presented as follows:

$$A^{\mathrm{T}}(-\nabla f)(A\beta') = \lambda\Gamma, \Gamma_i \in \begin{cases} \mathrm{sign}(\beta'), & \text{if } \beta' \neq 0; \\ [-1, 1], & \text{if } \beta' = 0 \end{cases} \tag{10}$$

where ∇f is the function of $\mathbb{R}^n$. We can define the equicorrelation set and the signs as follows:

$$\xi = \{i \in \{1, 2, \ldots, p\} : |A_i^{\mathrm{T}}(-\nabla f)(A\beta')| = \lambda\},$$
$$s = \mathrm{sign}(A_\xi^{\mathrm{T}}(-\nabla f)(A\beta')) \tag{11}$$

If $A \in \mathbb{R}^{\mathrm{nxp}}$ is a prediction matrix with entries drawn from the continuous probability distribution on $\mathbb{R}^{\mathrm{np}}$, then for a strictly convex function f, which is differentiable and $\lambda > 0$, the objective function in Formula (1) has a unique solution with probabilty as one. More generally, results can be applied to any convex differentiable loss function f, as in the following logistic regression loss as follows:

$$f(\omega) = \sum_{i=1}^{n}\{-y_i x_i + \log(1 + \exp(x_i))\} \tag{12}$$

Lemma 1: Local SGD can be defined as follows:

$$\omega_{t+1} = \frac{1}{N}\sum_{i=1}^{N}\omega_t - \eta_t\nabla f(\omega_t) \tag{13}$$

Theorem 1: Let Assumptions 1–3 hold. Then each round of convergence of regularized SGD with step size $\eta_t \leqslant 1/2l$ and variance σ^2 has the following properties:

$$E[f(\omega_t) - f(\omega_{t+1})] \geqslant E\left[\frac{\eta_t}{2}|\nabla f(\omega_t)^2|\right] - \eta_t\sigma^2 \tag{14}$$

The convergence results with variance as zero $\sigma = 0$ step size, $\eta_t = 1/2l$ is given by Ref. [73] as $\mathcal{O}(lB/T)$. The amount of variance affects convergence time. The larger the variance, the larger the convergence time to reach. Therefore, the use of elastic term remediates this effect by elastic optimization approach which improves the prediction accuracy and the training efficiency by reaching target accuracy in less number of communication rounds.

Theorem 2: Let Assumption 2 hold. That is, $E||\nabla F_j(\omega) - \nabla f(\omega)||^2 \leqslant \sigma^2$. For any ϵ, we have $\sqrt{1 + \frac{\sigma^2}{\epsilon}} \geqslant D_\epsilon$. We follow the assumption of dissimilarity as stated in Ref. [48]. We can derive the relationship between the gradient dissimilarity for the local objectives and the bounded variance for the gradients as follows:

$$E_j||\nabla F_j(\omega) - \nabla f(\omega)||^2 \leqslant \sigma^2 \tag{15}$$

We further expand Formula (15),

$$E_j||\nabla F_j(\omega) - \nabla f(\omega)||^2,$$
$$E_j||\nabla F_j(\omega_{t+1}) - \nabla f(\omega_{t+1})||^2,$$
$$E_j||\nabla F_j(\omega_{t+1}) - \nabla f(\omega_t) + \nabla f(\omega_t) - \nabla f(\omega_{t+1})||^2,$$
$$E_j||\nabla F_j(\omega_{t+1}) - \nabla f(\omega_t)| - |\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2.$$

The combined formula can be written as follows:

$$E_j||\nabla F_j(\omega) - \nabla f(\omega)||^2 \leqslant \sigma^2 + |\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2.$$

We define the dissimilarity B for $|\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2 \neq 0$ as the following formula:

$$B(\omega_{t+1}) = \sqrt{\frac{E_j[||\nabla F_j(\omega) - \nabla f(\omega)||^2]}{|\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2}} \tag{16}$$

$$B(\omega_{t+1})^2 = \frac{E_j||\nabla F_j(\omega) - \nabla f(\omega)||^2}{|\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2} \leqslant \frac{\sigma^2}{|\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2} + 1.$$

Theorem 2 and Assumption 2 quantify the dissimilarity that the devices can possess in the edge federated learning. The dissimilarity in local functions increases when $B(\omega) \geqslant 0$. So, the larger B means larger dissimilarity among edge devices. For all ω there exists ϵ such that $B(\omega) \leqslant B_\epsilon$. We can say that there is a sub-optimal ϵ solution which differs among the edge devices. When the gradients are bounded by some non-negative constants,we get the following implication:

$$B(\omega_{t+1}) = \sqrt{\frac{E_j[||\nabla F_j(\omega) - \nabla f(\omega)||^2]}{|\nabla f(\omega_{t+1}) - \nabla f(\omega_t)||^2}} \leqslant B_\epsilon \leqslant \sqrt{1 + \frac{\sigma^2}{\epsilon}} \tag{17}$$

Theorem 3 (ϵ-optimal stability): An algorithm $X(\cdot)$ is ϵ optimal stable if the datasets $D_1 = \{d_1, d_2, \ldots, d_n\}$ and its twin $D_2 = \{d_1', d_2', \ldots, d_n'\}$ with d being the data test sample have the following:

$$E_X[|f(X(D_1); d) - f(X(D_2; d))|] \leqslant \epsilon \tag{18}$$

Therefore, ϵ-stability indicates that the generalization is bounded by ϵ.

Theorem 4 (smooth functions): Let $F(\omega)$ be a function which is γ-smooth with step size, $\eta = \frac{\epsilon}{N^2\gamma}$ and T be the number of communication rounds. Then, after $\frac{2}{\epsilon^2}N^2\gamma(F(\omega) - F^*) \leqslant T$ updates with $(1 + \delta/2)\epsilon \geqslant E[||\nabla F(w_i)||_2^2]$, where $N^2 = 1/n \sum_{i=1}^{n} ||\nabla f_i(\omega)||_2^2$, ϵ-optimality condition, δ is a constant.

Theorems 3 and 4 have been also discussed in Ref. [69] which discussed the informal convergence guarantees for gradient diversity to achieve ϵ-optimal solution by bounding the distance to the optimal by B times.

Lemma 2: With the objective function in Formula (1) satisfying Assumptions 1–4. Let ω^t not be a stationary solution in proposed algorithm, then the global model objective is decreased as follows:

$$\mathrm{E}[f(\omega)^{t+1}] \leqslant f(\omega^t).$$

Theorem 5 (elastic convergence): Elastic net utilized two tuning parameter $\lambda_1, \lambda_2 \geqslant 0$. With $\lambda_2 > 0$, then the solution for elastic net is unique. We consider the fact that for any fixed $\lambda_1 > 0$, the elastic net converges to the minimum of L2 norm lasso solution. By fixing any input from X and $\lambda_1 > 0$, for almost every output predictor y, the elastic net converges to LARS lasso l_1 norm solution as $\lambda_2 \to 0^+$.

Proof: According to Lemma 13 defined in Ref. [74], if for any output vector, $y \notin N$, where $N \subseteq \mathbb{R}^n$, the LARS lasso satisfies $\beta'^{\mathrm{LARS}}(\lambda_1)_i \neq 0, \forall i \in \xi$. To fix $y \notin N$, we rewrite lasso solution as follows:

$$\beta'^{\mathrm{LARS}}_{-\xi}(\lambda_1) = 0,$$
$$\beta'^{\mathrm{LARS}}_{\xi}(\lambda_1) = (A_\xi^{\mathrm{T}}A_\xi)^+(A_\xi^{\mathrm{T}}y - \lambda_1 s) \tag{19}$$

where ξ is equicorrelation set, A is the predictor matrix, s is the sign, β' is lasso solution, β'^{LARS} is LARS lasso solution, and ϵ is optimal stable point. We define the function as

$$f(\lambda_2) = (A_\xi^{\mathrm{T}}A_\xi + \lambda_2 I)^{-1}(A_\xi^{\mathrm{T}}y - \lambda_1 s) \text{ for } \lambda > 0,$$
$$f(0) = (A_\xi^{\mathrm{T}}A_\xi)^+(A_\xi^{\mathrm{T}}y - \lambda_1 s) \tag{20}$$

with the equicorrelation correlation set ξ and fixed sign s, the function f is continuous on $[0, \infty)$. Therefore, for small $\lambda_2 > 0$, the elastic net solution for both the tuning parameters is given as follows:

$$\beta'^{\mathrm{Elastic}}_{-\xi}(\lambda_1, \lambda_2) = 0 \text{ and } \beta'^{\mathrm{Elastic}}_{-\xi}(\lambda_1, \lambda_2) = f(\lambda_2) \tag{21}$$

We show that the above solution satisfies Karush-Kuhn-Tucker (KKT) optimality conditions for lasso as in Ref. [74], for smaller values of λ_2. Therefore, the KKT conditions for the elastic net problem can be presented as follows:

$$A^{\mathrm{T}}(y - A\beta'^{\mathrm{Elastic}}) - \lambda_2\beta'^{\mathrm{Elastic}} = \lambda_1\Gamma \tag{22}$$

where Γ is the subgradient from the KKT optimality condition.

We have Γ for the i-th subgradient defined as follows:

$$\Gamma_i \in \begin{cases} \mathrm{sign}(\beta'^{\mathrm{Elastic}}), & \text{if } \beta'^{\mathrm{Elastic}} \neq 0; \\ [-1, 1], & \text{if } \beta'^{\mathrm{Elastic}} = 0 \end{cases} \tag{23}$$

Since, $f(0) = \beta'^{\mathrm{LARS}}(\lambda_1)$ (the equicorrelation coefficients of LARS lasso) at the tuning parameter λ_1 and $y \notin N$, there exists the continuity of f for small λ_2. Further, we know $||A_{-\xi}^{\mathrm{T}}(y - A_\xi f(0))||_\infty$. Therefore, by having direct calculations we get the following which verifies KKT conditions for small λ_2 with I being the identity matrix which does not have any inverse.

$$\begin{aligned} &A_\xi^{\mathrm{T}}(y - A_\xi f(\lambda_2)) - \lambda_2 f(\lambda_2) = \\ &A_\xi^{\mathrm{T}}y - (A_\xi^{\mathrm{T}}A_\xi + \lambda_2 I)(A_\xi^{\mathrm{T}}A_\xi + \lambda_2 I)^{-1}A_\xi^{\mathrm{T}}y + A_\xi^{\mathrm{T}}y - \\ &(A_\xi^{\mathrm{T}}A_\xi + \lambda_2 I)(A_\xi^{\mathrm{T}}A_\xi + \lambda_2 I)^{-1}\lambda_1 s = \lambda_1 s \end{aligned} \tag{24}$$

4 Experiment

In this section, we analyse the performance of our proposed algorithm, FedEN, in the presence of stragglers in edge federated learning. FedEN is described in Algorithm 2. Further, we verify the effectiveness of FedEN in case of statistical heterogeneity imposed due to partial results submission from edge devices. We also discuss the result of mixing parameter α in the local objectives of the participating edge devices. We then

Algorithm 2 FedEN: Elastic optimized federated learning

Input: B, N, E, and η
Output: Trained global model
Edge Device: $i = 1, 2, \ldots, N$:
target accuracy not achieved
Global model parameters downloaded to edge devices
Local model initialization and training using Eq. (8)
$y_i \leftarrow \omega$
$\omega_i^{t+1} \leftarrow \omega_i^t - \eta_{i\phi_i}(\omega_i^t) + \pi(\omega_i^t)$
if local training complete after E steps **then**
  send the updated model parameters from model y_i to the edge server
end if
Edge Server:
if local updates received by the edge server **then**
  perform aggregation using Eq. (5)
end if

compare FedEN with the benchmark algorithms, FedAvg and FedProx. For simplicity we consider federated training at one edge layer. Experiments are implemented with well-known deep learning framework, PyTorch[75] with python modules. The set up can be described as follows:

System heterogeneity: We consider that 10% and 50% of heterogeneous devices are stragglers and contribute their incomplete results in the model aggregation. Therefore, 10% of computational heterogeneity (alias to system heterogeneity), means the straggling devices are 10% out of total participating clients. Similarly, 50% of heterogeneity indicates that half of the participating edge devices are stragglers and the variance in data will be high causing divergence from the local objective.

Statistical heterogeneity: Our primary objective is to perform image classification using the datasets, MNIST and CIFAR-10. MNIST consists of 60 000 training samples and 10 000 testing samples. Whereas CIFAR-10 has 50 000 training samples and 10 000 testing samples. We assume the edge devices have half of the data from similar distribution. Hence, only half of the data of all the devices are not from the overall distribution. Therefore, we divide data between the edge devices in such a way that each device has some data from same distributions. In our case, each device will have some data from each of the 10 classes from MNIST and CIFAR-10 datasets. Further, we consider two sources of statistical heterogeneity contributing to emergence of stragglers. The incomplete results or partial results submission causes the edge devices to diverge from the local objectives and enhances the straggling-effect thereby contributing to the statistical heterogeneity. Similarly, the high dimensional data from the edge devices have large number of parameters which further add on to the data variance (alias to statistical heterogeneity). Hence, the former specified heterogeneity sources cause stragglers.

Training hyper-parameters: For training on image classification using MNIST and CIFAR-10, we use convolutional neural networks. The hyper-parameters used are listed in Table 2.

There are various machine learning models that can be used according to the problem statement and requirements. Different models include linear regression, logistic regression, support vector machines (SVM), neural networks (NN), and Bayesian regression. Any of the aforementioned models could be used as per the problem type and the solution needed. For us, we aim for image classification using MNIST and CIFAR-10 datasets. We use convolutional neural networks. Convolutional neural networks are used to train on the image classification tasks. Training is performed using MNIST and CIFAR-10 image datasets. For MNIST, local training is performed on training set. MNIST model network starts with the input layer followed by the two convolutional layers of size 5 pixel × 5 pixel. These convolutional layers are used for feature extraction and the linear layer at the end of the network acts as a classifier. Next, the convolutional layer is followed up with max pooling of size 2 pixel × 2 pixel. The max pooling layer helps in dimensionality reductions. After max pooling layer, a fully connected layer of 512 units is used. Activation function is used in ReLu. Finally, the output layer consists of Softmax function for predicting the classification probabilities. For CIFAR-10, CNN model has 64 (5 pixel × 5 pixel) filters for two convolutional layers. Next, convolutional layers are followed by two fully connected layers. First fully connected layer consists of 394 neurons and the second fully connected layer consists of 192 neurons. We use stochastic gradient descent (SGD) as the optimizer and the learning rate of 0.01. We adopt local mini-batch size of 10. The CNN structure specified earlier is similar to the one specified by Ref. [13]. Further, baseline algorithms (FedAvg and FedProx) also use the same CNN architecture (for fair comparison). The test accuracy for both MNIST and CIFAR-10 datasets is shown in Table 3 and the train losses are given in Table 4. Further, the bar chart representation for CIFAR-10 and MNIST accuracy is shown in Figs. 4–7 and their losses are shown in Figs. 8–11. Below are the federated learning algorithms which we use as the benchmarks to compare our proposed model.

Table 2 Training hyper-parameters.

Parameter	Value
Local epoch (E)	10
Local batch size (B)	10
Communication round (T)	100
Edge device (N)	100
Client selection fraction (C)	0.01
Learning rate (η)	0.01

- **FedAvg:** Local updation rule involves simple stochastic gradient descent for some number of iterations before communicating results with server.
- **FedProx:** Edge devices update their local model

Table 3 CIFAR-10/MNIST test accuracy with 10% and 50% stragglers.

Straggler number	Method	Test accuracy (%)							
		CIFAR-10				MNIST			
		R50	R100	R150	R200	R50	R100	R150	R200
10% straggler	FedAvg[13]	64.44	68.99	75.02	83.91	88.02	90.02	93.01	95.01
	FedProx[48]	67.80	75.01	78.99	86.9	89.9	91.5	94.5	96.09
	FedEN	71.94	78.9	85.5	**88.7**	92.7	94.89	96.83	**98.89**
50% straggler	FedAvg[13]	62.99	65.09	71.29	80.02	87.21	89.5	92.15	94.3
	FedProx[48]	65.80	73.99	76.02	83.50	88.57	89.55	92.41	95
	FedEN	69.94	82.52	83.89	**86.95**	91.81	92.89	95.98	**97.05**

Table 4 CIFAR-10/MNIST train loss with 10% and 50% stragglers.

Straggler number	Method	Train loss							
		CIFAR-10				MNIST			
		R50	R100	R150	R200	R50	R100	R150	R200
10% straggler	FedAvg	0.048	0.043	0.043	0.040	0.039	0.029	0.028	0.025
	FedProx	0.048	0.041	0.031	0.036	0.039	0.028	0.025	0.024
	FedEN	0.046	0.038	0.037	**0.034**	0.036	0.027	0.025	**0.022**
50% straggler	FedAvg	0.051	0.045	0.040	0.042	0.042	0.040	0.032	0.030
	FedProx	0.050	0.043	0.035	0.038	0.039	0.035	0.029	0.028
	FedEN	0.046	0.040	0.033	**0.035**	0.038	0.030	0.029	**0.025**

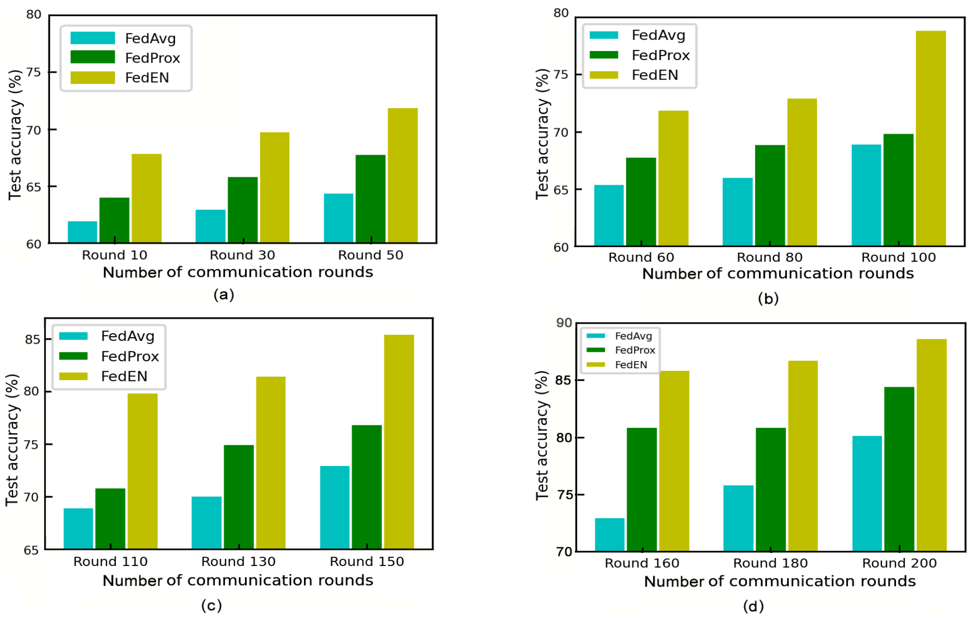


Fig. 4 CIFAR-10 test accuracy with 10% stragglers.

with a proximal term in their objective function. Here, we consider it as a benchmark since it also deals with the statistical heterogeneity via proximal term.

We compare the performance of our proposed model, FedEN, with such benchmarks in terms of different numbers of communication rounds taken to reach the target accuracy. We consider the partial device participation scenario where the edge devices contribute their partial results to the edge server since it cannot complete all training iterations. These devices are known to be stragglers with different computation capabilities. Hence, data heterogeneity also spikes due to this type of participation. We use MNIST (consisting of 60 000 train and 10 000 test samples) and CIFAR-10 (consisting of 50 000 train and 10 000 test samples) for classification task. We have implemented our algorithm in python using pyTorch consisting of different modules for local training, server training. Further, we incorporated 10% and 50% of system heterogeneity to replicate the real world uncertainty of straggling devices due to system

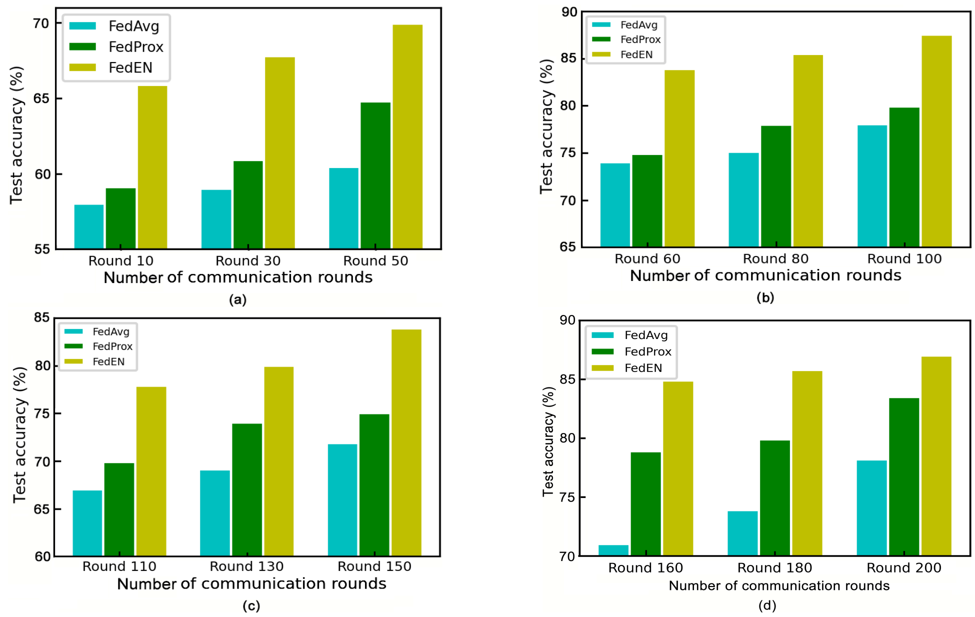


Fig. 5 CIFAR-10 test accuracy with 50% stragglers.

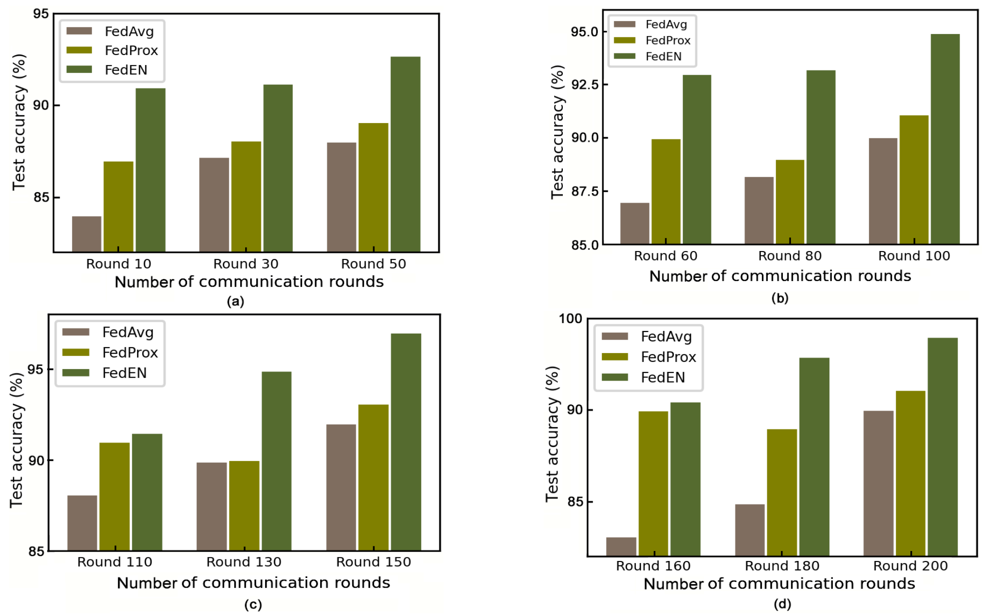


Fig. 6 MNIST test accuracy with 10% stragglers.

heterogeneity. For instance, as can be seen in Table 1, test accuracy for CIFAR-10 dataset with 10% of stragglers for 200 rounds is listed and visually demonstrated as bar graphs. The accuracy over the test data samples is demonstrated over communication rounds 50, 100, 150, and 200. Initially, around the communication round 50, the test accuracy for FedAvg is 64.44% and for FedProx it is 67.80%. FedEN achieves the accuracy of 71.94% for around 50 communication rounds. Similarly, considering round 100, FedEN attains the accuracy of 78.9%, higher than that of FedAvg and FedProx. Followed by the test accuracy of 85.5% as compared to FedAvg and FedProx which are 75.02% and 78.99%. Comparing with all previous training rounds, FedEN has always achieved a better accuracy than the benchmark algorithms. Lastly moving on the communication round 200 FedEN achieves an accuracy of 88.70%. Therefore, comparing with all communication rounds, FedEN achieves higher test accuracy for CIFAR-10 classification with consideration of 10% stragglers contributing to statistical heterogeneity.

5 Conclusion and Future Work

Our work proposes elastic optimized federated learning

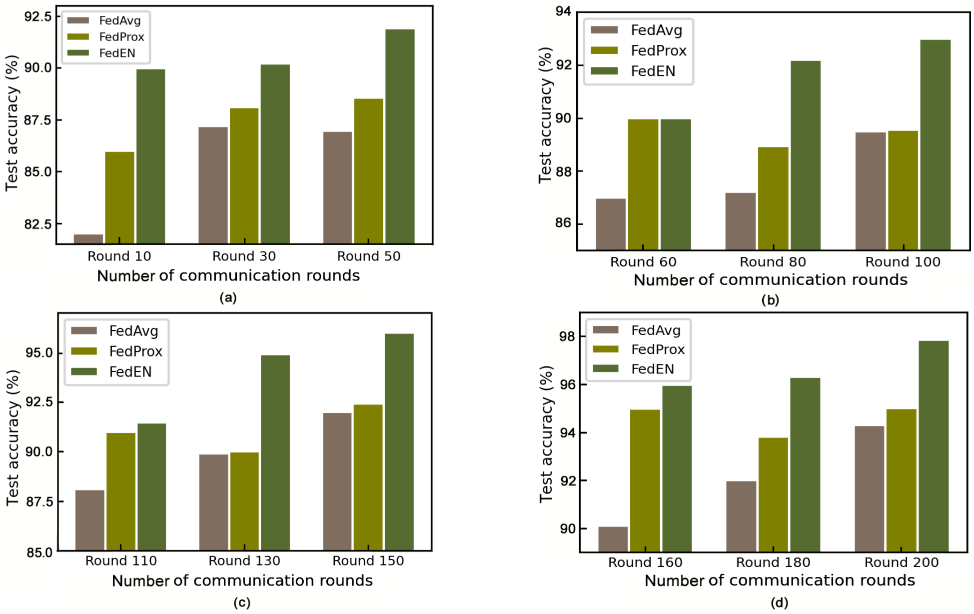


Fig. 7 MNIST test accuracy with 50% stragglers.

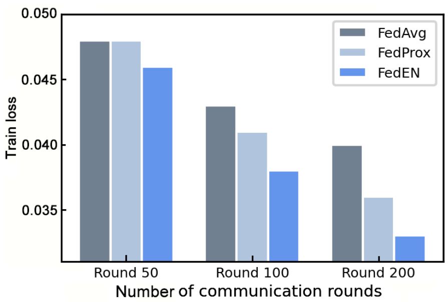


Fig. 8 CIFAR-10 train loss with 10% stragglers.

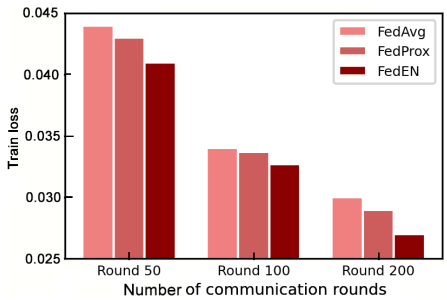


Fig. 10 MNIST train loss with 10% stragglers.

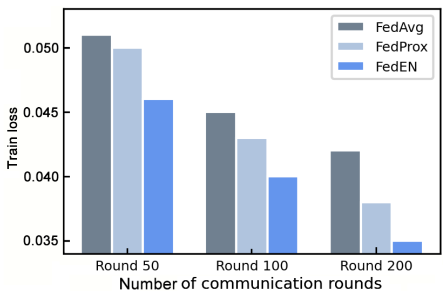


Fig. 9 CIFAR-10 train loss with 50% stragglers.

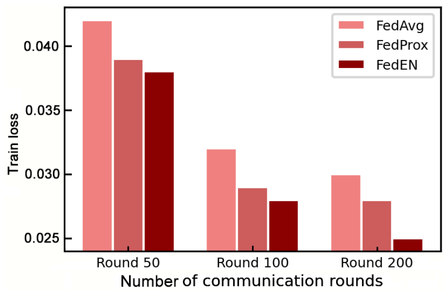


Fig. 11 MNIST train loss with 50% stragglers.

(FedEN) model. FedEN works for performance improvement for the edge devices federated training. Hence, local feature selection optimization problem and the gradient divergence caused by data heterogeneity can be solved thereby mitigating stragglers. We also prove the theoretical analysis of the proposed algorithm which improves the test accuracy over different number of communication rounds required for the model aggregation. Our experiments indicate that having elastic optimization in the local objectives of the participating edge devices can actually improve training performance by reducing the number of communication rounds required to achieve the target accuracy. Our proposed algorithm, elastic optimized federated learning (FedEN), performs better than benchmark algorithms such as FedAvg and FedProx by achieving better accuracy and less training loss. The proposed algorithm, FedEN, has the following advantages over the benchmarks in edge federated learning:

• FedEN allows data from stragglers to form the collaborative model where we set the percentage of stragglers as 10% and 50% of the edge devices during federated training which means that 10% or 50% of edge devices do not perform all training rounds.

• FedEN reduces the number of communication rounds to reach target accuracy compared with the benchmark algorithms

• A balance between a reduction in the number of parameters and their shrinkage makes the objective function smooth and improves prediction accuracy.

FedEN is the re-parameterization of FedAvg and FedProx such that the tuning parameter λ and the mixing parameter α can alter the training performance. When $\lambda = 0$, FedEN is similar to FedAvg. Furthermore, when the mixing parameter $\alpha = 0$, then FedEN is similar to FedProx where the proximal term is used in the local objectives. Stragglers due to large number of parameters from high dimensional data from IoT edge devices, for instance, cause the computation burden on the edge devices. Similarly, due to 10% and 50% of edge devices contributing the partial results, the statistical heterogeneity worsens thereby causing gradient divergence. Hence, FedEN can balance between the two penalization mentioned earlier, lasso and ridge, thereby producing sparse models with better prediction accuracy.

Limitations of FedEN: The main limitation of FedEN is the uncertainty about the actual percentage of stragglers that might exist during the training process. Additionally, the ad-hoc nature of wireless connectivity as well as system heterogeneity can cause an edge device which is working perfectly at first to become a straggler at any point in the training time or during any iteration round. We have only investigated model training with convolutional neural networks. Training on different neural networks has not been investigated which would open doors to address many training challenges for edge devices. In addition to this, cross-validation for tuning parameter could possibly be an option to further improve the training and testing results of the proposed algorithm.

References

[1] D. Pandey, H. Wang, X. Yin, K. Wang, Y. Zhang, and J. Shen, Automatic breast lesion segmentation in phase preserved DCE-MRIs, *Health Information Science and Systems*, vol. 10, p. 9, 2022.

[2] F. Zhang, Y. Wang, S. Liu, and H. Wang, Decision-based evasion attacks on tree ensemble classifiers, *World Wide Web*, vol. 23, no. 5, pp. 2957–2977, 2020.

[3] M. Peng, J. Zhu, H. Wang, X. Li, Y. Zhang, X. Zhang, and G. Tian, Mining event-oriented topics in microblog stream with unsupervised multi-view hierarchical embedding, *ACM Transactions on Knowledge Discovery from Data*, vol. 12, no. 3, pp. 1–26, 2018.

[4] J. Y. Li, Z. H. Zhan, H. Wang, and J. Zhang, Data-driven evolutionary algorithm with perturbation-based ensemble surrogates, *IEEE Transactions on Cybernetics*, vol. 51, no. 8, pp. 3925–3937, 2021.

[5] T. Huang, Y. J. Gong, S. Kwong, H. Wang, and J. Zhang, A niching memetic algorithm for multi-solution traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 508–522, 2019.

[6] Y. H. Zhang, Y. J. Gong, Y. Gao, H. Wang, and J. Zhang, Parameter-free Voronoi neighborhood for evolutionary multimodal optimization, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 335–349, 2020.

[7] S. Siuly, O. Alcin, E. Kabir, A. Sengur, H. Wang, Y. Zhang, and F. Whittaker, A new framework for automatic detection of patients with mild cognitive impairment using resting-state EEG signals, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 9, pp. 1966–1976, 2020.

[8] J. Y. Li, K. J. Du, Z. H. Zhan, H. Wang, and J. Zhang, Distributed differential evolution with adaptive resource allocation, *IEEE Transactions on Cybernetics*, doi: 10.1109/TCYB.2022.3153964.

[9] J. Yin, M. J. Tang, J. Cao, H. Wang, M. You, and Y. Lin, Vulnerability exploitation time prediction: An integrated framework for dynamic imbalanced learning, *World Wide Web*, vol. 25, pp. 401–423, 2021.

[10] H. Wang, Y. Wang, T. Taleb, and X. Jiang, Editorial: Special issue on security and privacy in network computing, *World Wide Web*, vol. 23, pp. 951–957, 2019.

[11] H. Wang, L. Sun, and E. Bertino, Building access control policy model for privacy preserving and testing policy conflicting problems, *Journal of Computer and System Sciences*, vol. 80, no. 8, pp. 1493–1503, 2014.

[12] X. Sun, H. Wang, J. Li, and Y. Zhang, Injecting purpose and trust into data anonymisation, *Computers & Security*, vol. 30, no. 5, pp. 332–345, 2011.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Proc. 20^{th} International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[14] J. Yin, M. Tang, J. Cao, M. You, H. Wang, and M. Alazab, Knowledge-driven cybersecurity intelligence: Software vulnerability co-exploitation behavior discovery, *IEEE Transactions on Industrial Informatics*, doi: 10.1109/TII.2022.3192027.

[15] X. Sun, H. Wang, J. Li, and J. Pei, Publishing anonymous survey rating data, *Data Min. Knowl. Discov.*, vol. 23, pp. 379–406, 2011.

[16] Y. F. Ge, M. Orlowska, J. Cao, H. Wang, and Y. Zhang, MDDE: Multitasking distributed differential evolution for privacy-preserving database fragmentation, *The VLDB Journal*, vol. 31, pp. 957–975, 2022.

[17] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, A blockchained federated learning framework for cognitive computing in industry 4.0 networks, *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2021.

[18] Y. Qu, L. Gao, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, Decentralized privacy using blockchain-enabled federated learning in fog computing, *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, 2020.

[19] L. Sun, J. Ma, H. Wang, Y. Zhang, and J. Yong, Cloud service description model: An extension of USDL for cloud services, *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 354–368, 2015.

[20] K. Cheng, L. Wang, Y. Shen, H. Wang, Y. Wang, X. Jiang, and H. Zhong, Secure k-NN query on encrypted cloud data with multiple keys, *IEEE Transactions on Big Data*, vol. 7, no. 4, pp. 689–702, 2017.

[21] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, EdgeFed: Optimized federated learning based on edge computing, *IEEE Access*, vol. 8, pp. 209191–209198, 2020.

[22] X. Mo and J. Xu, Energy-efficient federated edge learning with joint communication and computation design, *Journal of Communications and Information Networks*, vol. 6, no. 2, pp. 110–124, 2021.

[23] T. Zhou, X. Li, C. Pan, M. Zhou, and Y. Yao, Multi-server federated edge learning for low power consumption wireless resource allocation based on user QoE, *Journal of Communications and Networks*, vol. 23, no. 6, pp. 463–472, 2021.

[24] J. Mills, J. Hu, and G. Min, Multi-task federated learning for personalised deep neural networks in edge computing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630–641, 2022.

[25] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network, *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2021.

[26] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, Federated multiagent actor–critic learning for age sensitive mobile-edge computing, *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1053–1067, 2022.

[27] S. R. Pandey, M. N. H. Nguyen, T. N. Dang, N. H. Tran, K. Thar, Z. Han, and C. S. Hong, Edge-assisted democratized learning toward federated analytics, *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 572–588, 2022.

[28] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. -C. Liang, Q. Yang, D. Niyato, and C. Miao, Federated learning in mobile edge networks: A comprehensive survey, *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[29] L. Gao, T. H. Luan, B. Gu, Y. Qu, and Y. Xiang, An introduction to edge computing, in *Privacy-Preserving in Edge Computing*, L. Gao, T. H. Luan, B. Gu, Y. Qu, and Y. Xiang, eds. Singapore: Springer, 2021, pp. 1–14.

[30] L. Gao, T. H. Luan, B. Gu, Y. Qu, and Y. Xiang, Blockchain based decentralized privacy preserving in edge computing, in *Privacy-Preserving in Edge Computing*, L. Gao, T. H. Luan, B. Gu, Y. Qu, and Y. Xiang, eds. Singapore: Springer, 2021, pp. 83–109.

[31] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, Adaptive federated learning in resource constrained edge computing systems, *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[32] M. E. Kabir, H. Wang, and E. Bertino, A role-involved purpose-based access control model, *Information Systems Frontiers*, vol. 14, pp. 809–822, 2012.

[33] H. Wang, Y. Zhang, J. Cao, and V. Varadharajan. Achieving secure and flexible M-services through tickets, *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 33, no. 6, pp. 697–708, 2003.

[34] M. E. Kabir, A. N. Mahmood, H. Wang, and A. K. Mustafa, Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing, *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 408–417, 2015.

[35] B. Gu, L. Gao, X. Wang, Y. Qu, J. Jin, and S. Yu, Privacy on the edge: Customizable privacy-preserving context sharing in hierarchical edge computing, *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2298–2309, 2020.

[36] M. M. A. Aziz, M. M. Anjum, N. Mohammed, and X. Jiang, Generalized genomic data sharing for differentially private federated learning, *Journal of Biomedical Informatics*, vol. 132, p. 104113, 2022.

[37] H. Wang, Z. Kaplan, D. Niu, and B. Li, Optimizing federated learning on non-IID data with reinforcement learning, in *Proc. IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, Toronto, Canada, pp. 1698–1707, 2020.

[38] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, A survey of machine and deep learning methods for internet of things (IoT) security, *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[39] Z. Tang, H. Hu, and C. Xu, A federated learning method for network intrusion detection, *Concurrency and Computation*: *Practice and Experience*, vol. 34, no. 10, p. e6812, 2022.

[40] J. Mills, J. Hu, and G. Min, Client-side optimisation strategies for communication-efficient federated learning, *IEEE Communications Magazine*, vol. 60, no. 7, pp. 60–66, 2022.

[41] Q. Qi and X. Chen, Robust design of federated learning for edge-intelligent networks, *IEEE Transactions on Communications*, vol. 70, no. 7, pp. 4469–4481, 2022.

[42] P. Tian, W. Liao, W. Yu, and E. Blasch, WSCC: A weight similarity based client clustering approach for non-IID federated learning, *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20243–20256, 2022.

[43] Z. Ji, L. Chen, N. Zhao, Y. Chen, G. Wei, and F. R. Yu, Computation offloading for edge-assisted federated learning, *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9330–9344, 2021.

[44] H. Liu, X. Yuan, and Y. J. A. Zhang, Reconfigurable intelligent surface enabled federated learning: A unified communication-learning design approach, *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7595–7609, 2021.

[45] Z. Lin, H. Liu, and Y. J. A. Zhang, Relay-assisted cooperative federated learning, *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7148–7164, 2022.

[46] Z. Zhou, Y. Li, X. Ren, and S. Yang, Towards efficient and stable k-asynchronous federated learning with unbounded stale gradients on non-IID data, *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3291–3305, 2022.

[47] T. Nishio and R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in *Proc. ICC 2019-2019 IEEE International Conference on Communications* (*ICC*), Shanghai, China, 2019, pp. 1–7.

[48] 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.

[49] Y. -S. Jeon, M. M. Amiri, J. Li, and H. V. Poor, A compressive sensing approach for federated learning over massive MIMO communication systems, *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1990–2004, 2021.

[50] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, TiFL: A tier-based federated learning system, in *Proc. 29th International Symposium on High-Performance Parallel and Distributed Computing*, Stockholm, Sweden, 2020, pp. 125–136.

[51] Y. Liu, Y. Qu, C. Xu, Z. Hao, and B. Gu, Blockchain-enabled asynchronous federated learning in edge computing, *Sensors*, vol. 21, no. 10, p. 3335, 2021.

[52] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, Revisiting distributed synchronous SGD, arXiv preprint arXiv: 1604.00981, 2016.

[53] N. Ferdinand, H. Al-Lawati, S. C. Draper, and M. Nokleby, Anytime minibatch: Exploiting stragglers in online distributed optimization, arXiv preprint arXiv: 2006.05752, 2020.

[54] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, Gradient coding: Avoiding stragglers in distributed learning, in *Proc. 34th International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 3368–3376.

[55] E. Yang, D. K. Kang, and C. H. Youn, BOA: Batch orchestration algorithm for straggler mitigation of distributed DL training in heterogeneous GPU cluster, *The Journal of Supercomputing*, vol. 76, pp. 47–67, 2020.

[56] Q. Zhou, S. Guo, H. Lu, L. Li, M. Guo, Y. Sun, and K. Wang, Falcon: Addressing stragglers in heterogeneous parameter server via multiple parallelism, *IEEE Transactions on Computers*, vol. 70, no. 1, pp. 139–155, 2020.

[57] R. Bitar, M. Wootters, and S. E. Rouayheb, Stochastic gradient coding for straggler mitigation in distributed learning, *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 277–291, 2020.

[58] S. Prakash, S. Dhakal, M. R. Akdeniz, Y. Yona, S. Talwar, S. Avestimehr, and N. Himayat, Coded computing for low-latency federated learning over wireless edge networks, *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 233–250, 2020.

[59] W. Wu, L. He, W. Lin, and R. Mao, Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems, *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1539–1551, 2020.

[60] Z. Li, H. Zhou, T. Zhou, H. Yu, Z. Xu, and G. Sun, ESync: Accelerating intra-domain federated learning in heterogeneous data centers, *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2261–2274, 2020.

[61] J. He, S. Guo, M. Li, and Y. Zhu, AceFL: Federated learning accelerating in 6G-enabled mobile edge computing networks, *IEEE Transactions on Network Science and Engineering*, doi: 10.1109/TNSE.2022.3190330.

[62] C. You, D. Feng, K. Guo, H. H. Yang, and T. Q. S. Quek, Semi-synchronous personalized federated learning over mobile edge networks, arXiv preprint arXiv: 2209.13115, 2022.

[63] T. Q. Dinh, D. N. Nguyen, D. T. Hoang, T. V. Pham, and E. Dutkiewicz, In-network computation for large-scale federated learning over wireless edge networks, *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2022.3190260.

[64] S. Singh, R. Sulthana, T. Shewale, V. Chamola, A. Benslimane, and B. Sikdar, Machine-learning-assisted security and privacy provisioning for edge computing: A survey, *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 236–260, 2022.

[65] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[66] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv: 1610.05492, 2016.

[67] Y. Zhang, H. Qu, D. Metaxas, and C. Chen, Local regularizer improves generalization, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 6861–6868, 2020.

[68] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, Federated learning: Challenges, methods, and future directions, *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[69] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett, Gradient diversity: A key ingredient for scalable distributed learning, in *Proc. Twenty-First International Conference on Artificial Intelligence and Statistics*, Playa Blanca, Spain, 2018, pp. 1998–2007.

[70] H. T. Nguyen, V. Sehwag, S. Hosseinalipour, C. G. Brinton, M. Chiang, and H. V. Poor, Fast-convergent federated learning, *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 201–218, 2020.

[71] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, Adaptive federated learning in resource constrained edge computing systems, *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[72] 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.

[73] G. Qu and N. Li, Accelerated distributed nesterov gradient

descent, *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2566–2581, 2019.

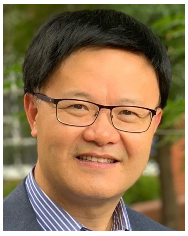
[74] R. J. Tibshirani, The lasso problem and uniqueness, *Electronic Journal of Statistics*, vol. 7, pp. 1456–1490, 2013.

[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, in *Proc. 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 8026–8037.

Khandakar Ahmed received the PhD degree from RMIT in 2015 and the MSc degree in networking and e-business centred computing (NeBCC) from the joint consortia of the University of Reading, UK, Aristotle University of Thessaloniki, Greece, and Charles III University of Madrid (UC3M), Spain in 2011. He started his career as a full-stack developer and later switched to the academy by joining his alma mater as a lecturer in 2007, where he completed his undergraduate. He taught at several universities in Europe and Australia and worked as a postdoctoral researcher at RMIT in 2015 and 2016 before joining Victoria University in 2017. Currently, he is a senior lecturer in IT at Victoria University. He has extensive industry engagement as a chief investigator in multiple research projects related to the Internet of Things, smart cities, machine learning, cybersecurity, and biomedical informatics. He received substantial industry funding over the last five years collaborating with industries and local, state, and federal governments in solving contemporary social problems using intelligent technologies.

Bruce Gu received the BInfoTech and MIT degrees from Deakin University, Australia in 2009 and 2011, respectively, and the PhD degree in computer science from Deakin University, Australia in 2020. He is currently a scientist in Shandong Computer Science Center (National Supercomputer Center in Jinan), China. Before joining Shandong Computer Science Center, he was a lecturer in the Discipline of Information Technology, College of Engineering and Science, Victoria University. He has over 10 years of industry engagement experience in Australia. His research interests include artificial intelligence, cybersecurity, privacy preserving, blockchain, Internet of Things, and edge AI. He has served as the general co-chair, TPC chair, and TPC members for many international conferences. He also serves as a reviewer for many top quality journals.

Khadija Sultana received the master degree in ICT from Latrobe University, Australia in 2020. She is currently pursuing the master degree in science at Victoria University, Melbourne, Australia. She has recently stepped in corporate industry and is currently working as a data analyst. Her research interests include data privacy, cyber security, federated learning technology, and their applications.



Hua Wang received the PhD degree from University of Southern Queensland, Australia. He is now a full time professor at Victoria University. He was a professor at University of Southern Queensland before he joined Victoria University. He has more than ten years teaching and working experience in applied informatics at both enterprise and university. He has expertise in electronic commerce, business process modeling, and enterprise architecture. He is a senior member of IEEE. As an chief investigator, six Australian Research Council Discovery grants have been awarded since 2006, and 350 peer reviewed scholar papers have been published. 16 PhD students have already graduated under his principal supervision.