CAAI Transactions on Intelligence Technology

Research Article

Decentralised federated learning with adaptive partial gradient aggregation

ISSN 2468-2322
Received on 26th March 2020
Revised on 23rd April 2020
Accepted on 30th April 2020
doi: 10.1049/trit.2020.0082
www.ietdl.org

Jingyan Jiang, Liang Hu ✉

College of Computer Science and Technology, Jilin University, Changchun 130012, People's Republic of China
✉ *E-mail: hul@jlu.edu.cn*

Abstract: Federated learning aims to collaboratively train a machine learning model with possibly geo-distributed workers, which is inherently communication constrained. To achieve communication efficiency, the conventional federated learning algorithms allow the worker to decrease the communication frequency by training the model locally for multiple times. Conventional federated learning architecture, inherited from the parameter server design, relies on highly centralised topologies and large nodes-to-server bandwidths, and convergence property relies on the stochastic gradient descent training in local, which usually causes the large end-to-end training latency in real-world federated learning scenarios. Thus, in this study, the authors propose the adaptive partial gradient aggregation method, a gradient partial level decentralised federated learning, to tackle this problem. In FedPGA, they propose a partial gradient exchange mechanism that makes full use of node-to-node bandwidth for speeding up the communication time. Besides, an adaptive model updating method further reduces the convergence rate by adaptive increasing the step size of the stable direction of gradient descent. The experimental results on various datasets demonstrate that the training time is reduced up to 14 × compared to baselines without accuracy degrade.

1 Introduction

Recent years have witnessed a significant improvement in the Internet of Things. Edge/Remote devices, such as phones, vehicles, and wearable sensors are connected with networks and generate a wealth of data each day. Federated learning [1–3] has emerged as an attractive paradigm to take advantage of decentralised data. In such settings, the goal is to learn a global shared deep neural network (DNN) model using distributed data. Different from conventional distributed machine learning, the participants in federated learning are connected across the wide area network (WAN), where the bandwidth constraints, and statistical heterogeneity in the user datasets present significant challenges. Of current federated learning solvers, FedAvg [1] has become the de facto scheme for non-convex federated learning. As illustrated in Fig. 1*a*, the central server selects a subset of all devices to send the global shared model to devices; after that, the device conducts multiple stochastic gradient descent (SGD) using local samples based on the received shared model. Then, the server randomly selects a subset of devices to upload their local updates. The global model is updated based on the averaging of received updates, and sends to the selected devices.

Although the local updating and low participation of FedAvg reduce the communication overhead, the end-to-end training time still faces the inevitable latency in network bottleneck of centralised server and the convergence rate of SGD. The millions of over WAN devices involved share one or a set of central servers to exchange local updates, leading to the non-negligible communication latency in exchanging updates. Mostly, the high latency is caused by the following reasons:

(i) *Scarce WAN bandwidth:* The devices in federated learning are geo-distributed, that is, the data is transmitted over WAN. However, as measured in [4, 5], the WAN bandwidth is a quite scarce resource. First, the bandwidth within a data centre is 15 × larger than the WAN bandwidth on average, 60 × in the worst case. Secondly, the WAN bandwidth is different significantly between different regions, i.e. up to 12 × difference. Thirdly, the WAN bandwidth varies over time. The large variance is larger than 4 × in a day. The unbalanced and scare WAN bandwidth may cause a high latency in federated learning.
(ii) *Bottleneck in centralised topology*: In FedAvg, all the chosen devices have to transmit the updates to the central server at each iteration (for a synchronised scheme). Network congestion often occurs on the server-side. Although it could set more servers for scaling, the congestion is not radically solved. The congestion may slow down the transmission time of updates.
(iii) *Large DNN models:* To achieve higher performance in accuracy, the DNNs' models become larger and larger, i.e. the model size of BERTLARGE, which is the state-of-the-art model in NLP, can be up to 1360 MB. Besides, the primal goal of federated learning is to train a large model using more decentralised data. Obviously, the large data size needs more time to transmit.

Thus, to further decrease the end-to-end training time, some advanced decentralised optimisation methods [6, 7] have been proposed, instead of All-Reduce [8] scheme, devices send local updates to only one or a group of selected devices. In real-world federated learning scenarios, the network capacities between nodes are highly uniformly distributed and smaller than that in a datacenter [9]. Thus, it is still extremely bandwidth costly when workers send the *full* model updates (e.g. the size can be up to 1360 MB in $BERT_{LARGE}$ [10]). To exchange partial updates is an intuitive way to tackle the network unbalance and bottleneck. Ako method in [11] exchanges partial gradient with all the peer devices, while Combo method in [12] exchanges partial model weights instead. Although these methods reduce the communication time, their simple averaging scheme still faces the convergence rate limitation of SGD.

To address the problems above, in this paper, we proposed novel decentralised, federated learning design as shown in Fig. 1*b*, introducing an adaptive partial gradient aggregation scheme, which not only makes full utilisation of sufficient node-to-node bandwidth by transmitting accumulated local gradient slice, called *Partial τ-Difference Gradients* in a peer-to-peer manner but also takes advantage of 'Adam' algorithm to speed up the convergence

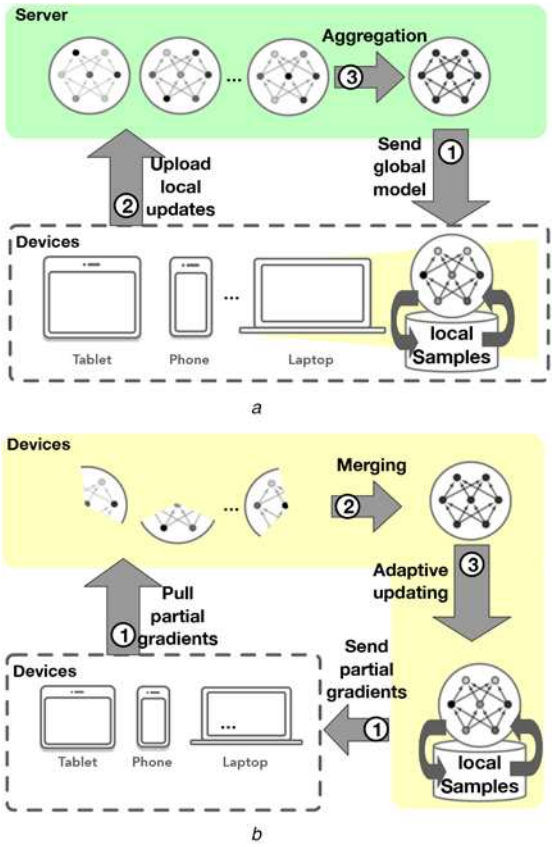

Fig. 1 *Federated learning architecture*

a FedAvg
b FedPGA

rate. In particular, the details of the design and the contributions are summarised as follows:

- First, we propose a decentralised partial gradient exchange mechanism. We 'split' a gradient into a set of slices – subsets that contain the same number of gradients that are not overlapped with each other. Devices perform slice level updates by aggregating a local slice with the corresponding slice from k other peer devices and merging into a full size mixed gradient.
- Secondly, to speed up the convergence rate further, we propose an adaptive updating method, which borrows the idea from the 'Adam' algorithm to adaptively set the learning rate according to the stability of the gradient direction. Besides, instead of the 'one-step' gradient, our 'Adam-like' method utilises the accumulated local gradient, called *τ-Difference Gradients*, to capture the cumulative gradient direction.
- Finally, we implement our adaptive partial gradient aggregation strategy into a prototype called FedPGA, and design experiments to evaluate its performance (in the non-i.i.d settings). Our results show that our design significantly reduces (upto 14 ×) the training time in practical network topology and bandwidth setup, without accuracy degrade.

2 Related work

Many distributed optimisation methods have been proposed to overcome the challenges in large scale distributed machine learning, such as primal methods [13–15] and primal-dual methods [16–18], while, in federated learning settings, the communication cost often becomes dominant compared to the computation cost.

By introducing the *local updating* and *partial devices participating* to balance the communication and computation in the large networks [1, 3, 19–22], these methods have shown significant improvements over traditional distributed optimisation approaches. For instance, Smith *et al*. [3] proposed a communication-efficient primal-dual optimisation method that trains separate but related models for each client and captures the relationship among clients through their relationship matrix. Although it has a theoretical convergence guarantee, it still faces the challenge of non-convex objectives, e.g. deep learning. FedAvg [1] tackles the non-convex problem by averaging local SGD updates, and has been shown to work well empirically. However, it suffers slow convergence when the training data is not identically distributed [23] and the centralised architecture will bring the network congestion in the global server. To reduce the network congestion, the decentralised methods [11, 12, 24] are proposed, every device is connected over WAN, and there is no global server to aggregate the updates. The former work exchanged the partial gradients with all the other devices, while the method in [12] exchanges the partial model weights with a subset of total devices. Jiang *et al*. [24] further proposed a bandwidth-aware device selection method to reduce the communication latency. Although their improvement in communication efficiency, these methods are based on the local updating (SGD) and simply global averaging, leading to the limits of the convergence rate of SGD.

To break the limits of local updating, Leroy *et al*. [25] proposed *adaptive averaging* methods inspired by Adam optimiser to reduce the number of communication rounds required. However, in this paper, we propose a general framework focus on optimisation in aggregation by exchanging average cumulative gradients instead of parameters or gradients.

3 Proposed framework: FedPGA

In this section, we first formally define the classical federated learning objective and methods, and describe the details of our proposed method, FedPGA. Then we analyse the performance of communication efficiency.

3.1 Preliminaries: federated learning

Federated learning aims to collaborate the samples from a large amount of geo-distributed devices (i.e. hundreds to millions), and communicate with a parameter server periodically to train a shared global model. Instead of exchanging private data, devices exchange the local updates for protecting data privacy. Formally, the objective function of federated learning is to minimise the following function:

$$\min_{w} F(w) = \sum_{k=1}^{K} p_k F_k\left(w; \mathcal{D}_k\right), \tag{1}$$

where K is the total number of devices, w is the global model weights, $p_k \geq 0$ and $\sum_k p_k = 1$. We could set p_k to be n_k/n, where $n = \sum_k n_k$ is the total number of samples of all the devices. The F_k is the local objective function of device k on its local available dataset $\mathcal{D}_k$, i.e. the empirical risk could be used: $F_k = \left(1/n_k\right) \sum_{j_k=1}^{n_k} f_{j_k}\left(w; \mathcal{D}_k\right)$, where $n_k = \left|\mathcal{D}_k\right|$ is the number of total samples on device k.

To solve the above problem (1), the main optimisation methodology is parallel SGD in distributed machine learning, where the model is updated iteratively using average gradients of the workers, as follows:

$$w_{t+1} \leftarrow w_t - \eta \sum \frac{n_k}{n} g_t^k, \tag{2}$$

where w_{t+1} and w_t are the model weights at the $(t+1)$th and tth iterations, respectively. η is a constant of step size, and $g_t^k = \nabla_{w_t} F_k\left(w_t; \mathcal{D}_k\right)$ is an one-step stochastic gradient of the objective function evaluated on the mini-batch of dataset for device k.

Despite the ease of implementation, the parallel SGD suffers from the large communication overhead in practice settings in terms of *rounds of communications* and *the amount of data to exchange*. In

Input: $K, T, \eta, E, w_0, N, n, n_k, p_k, k = 1, ..., N, \tau$

Server executes:
for $t = 0, ..., T-1$ **do**
  server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  send w_t to all chosen devices
  for each device $k \in S_t$ in parallel **do**
    $w_{t+1}^k \leftarrow \texttt{ClientUpdatae}(k, w_t, \eta, \tau)$
  end for
  $w_{t+1} \leftarrow \sum_{k \in S_t} p_k w_{t+1}^k$
end for

$\texttt{ClientUpdatae}(k, w_t, \eta, \tau)$ //Run on device k
  device k updates w_t for τ iterations of SGD on F_k with step size η to obtain w_{t+1}
return w_{t+1}

Fig. 2 *Algorithm 1: Federated averaging (FedAvg)*

order to reduce the rounds of communications, *Local SGD* approaches [19, 26, 27] increase the interval of global aggregation, that is, each device conducts multiple SGD using local data before global model synchronisation. To further decrease the communication overhead in federated learning settings, *FedAvg* [1], the leading algorithm, has demonstrated its empirical performance. it selects a subset S_t of devices to aggregate local updates (model weights) at each global synchronisation rounds, and the details are summarised as Algorithm 1 (see Fig. 2).

Although FedAvg shows good convergence in practice and theory, it suffers from the communication bottleneck because of the resource allocation problem in an inherent centralised architecture. We will discuss the limitation of centralised architecture in communication and the simple averaging aggregation.

3.2 FedPGA methods

Instead of a centralised architecture, we consider a decentralised solution, the network topology with N devices, each device has a network connection among the other $N-1$ devices. Instead of training a shared global model, each device trains its own model in the decentralised federated learning. The goal of each device i is to minimise the following objective function $F_i(w_i)$:

$$\min_{w_i} F_i(w_i) = \sum_{k=1}^{N} p_k F_k(w_k; \mathcal{D}_k). \tag{3}$$

To solve the problem (3), and reduce the end-to-end latency, we propose *partial gradient exchange* and *adaptive aggregation* scheme. Next, we will present the details.

3.2.1 Partial gradient exchange: Different from FedAvg, the devices in FedPGA exchange *partial gradients* rather than full model weights. The partial gradient exchange consists of two schemes, pulling and merging.

We suppose that each local update (We use gradient in this paper) splits into S slices. Each slice is called a *Partial Gradient*. Formally, it is defined as follows:

Definition 1: (Partial Gradients): We define a full gradient is g_t^i for device i at communication rounds t. We split the g_t^i into S slices, each slice $s \in [S]$ is called a partial gradients and denoted as $g_t^i(s)$

$$g_t^i := \left(g_t^i(1), g_t^i(2), ..., g_t^i(S)\right). \tag{4}$$

Device i uniform randomly selects a subset $\mathcal{S}_i$ of devices, where $|\mathcal{S}_i| = S$. The k th chosen device provides k th partial gradient $g_t^k(k)$, $k \in \mathcal{S}_i$.

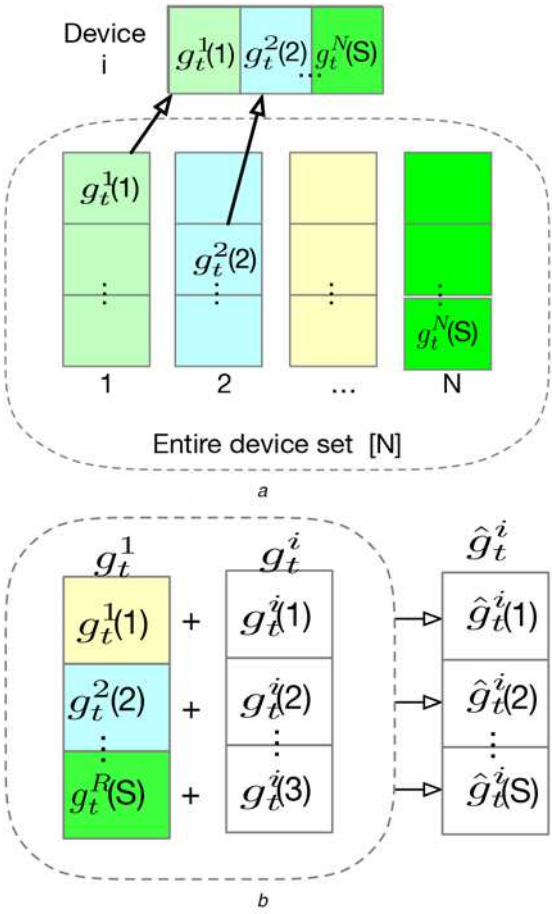

Fig. 3 *Partial gradients merging*

a Partial gradients
b Mixed gradients

Fig. 3*a* illustrates the pulling procedure which we name it *partial gradients pulling*. In the aggregation phase, the device needs to receive the model update from others. While the FedAvg requires the device to collect the whole model updates, partial gradients pulling allows the device to pull a different slice of the updates from different devices and rebuild a mixed update for aggregation.

After pulling the partial gradients, each device i will merge the partial gradients into the mixed gradient, as shown in Fig. 3*b*.

Definition 2: (Mixed Partial Gradients): We define a mixed partial gradient is $\hat{g}_t^i(s)$ for device i at communication rounds t at s th slice. The mixed partial gradient is the weighted averaging of received partial gradient $\hat{g}_{t,\text{recv}}^i(s)$ and local partial gradient $\hat{g}_{t,\text{local}}^i(s)$ at sth slice

$$\hat{g}_t^i(s) := \frac{|D_{\text{recv}}|\hat{g}_{t,\text{recv}}^i(s) + |D_{\text{local}}|\hat{g}_{t,\text{local}}^i(s)}{|D_{\text{recv}}| + |D_{\text{local}}|} \tag{5}$$

where $|D_{\text{recv}}|$ and $|D_{\text{local}}|$ are the number of samples of the pulled peer device and the local device.

Definition 3: (Mixed Gradients): We define a mixed gradient is $\hat{g}_t^i(s)$ for device i at communication rounds t. The mixed gradient is jointed by S mixed paritial gradients in order

$$\hat{g}_t^i := \left(\hat{g}_t^i(1), \hat{g}_t^i(2), \ldots, \hat{g}_t^i(S)\right). \tag{6}$$

After the merging phase, each device will conduct an adaptive updating, and we will present the details in the next section.

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 3, pp. 230–236

3.2.2 Adaptive updating: FedAvg relies on the 'delayed communication' to reduce the network overhead and end-to-end training latency. The number of communication rounds in FedAvg to achieve convergence is determined by local training results, and FedAvg is extremely vulnerable to the devices' data distributions. To solve the problem, we use the idea of adaptive gradient from RMSProp and Adam [28] and propose an adaptive updating method for federated learning, which accelerates the training with 'adaptive updating' instead of simple model averaging. Next, we present the detailed design of FedPGA in a bottom-up way.

Next, we present our adaptive updating approach. In our design, we propose a quasi-Adam aggregator on the top of FedAvg to boost the aggregation quality. Intuitively, in the SGD algorithm, the gradient provides descent direction for updating the model, and the learning rate controls the descent speed. The Adam algorithm achieves fast convergence by adaptively setting the learning rate according to the stability of the gradient direction. For example, if the gradient does not change much for a few iterations, the algorithm knows a larger learning rate can be applied. On the contrary, the learning rate decreases if the gradient fluctuates drastically.

Definition 4: (τ-Difference Gradient): We denote δ_t^i as the weighted difference of local model weights after τ times SGD for device i at communication rounds t

$$\delta_t^i := \frac{1}{\eta}\left(w_t^{i(\tau)} - w_t^{i(0)}\right) \tag{7}$$

where $w_t^{i(0)}$ and $w_t^{i(\tau-1)}$ are the model weights at the beginning SGD training and the τ th SGD training for device i at communication rounds t. η is the learning rate of SGD. Notice that, $w_t^{i(0)}$ is equals to w_t^i.

From (7), we could find out that we treat the τ-Difference Gradient δ_t^i as a 'gradient'. It is the update contributed by the device to be applied to w_t^i, which also indicates the descent direction and the learning rate for updating the model.

As we discussed in the previous section, the devices exchange the partial gradient. Further, combining the insight of τ-Difference Gradient δ_t^i, the devices will exchange the partial τ-Difference Gradient. After the device merges the partial τ-Difference Gradient and reconstructs them into a mixed τ-Difference Gradient $\hat{\delta}_t^i$. FedPGA then processes it in an Adam manner as follows:

$$\begin{aligned} u_t &\leftarrow \beta_1 u_{t-1} + (1-\beta_1)\hat{\delta}_t^i \\ v_t &\leftarrow \beta_2 v_{t-1} + (1-\beta_2)(\hat{\delta}_t^i)^2 \\ \hat{u}_t &\leftarrow \frac{u_t}{1-(\beta_1)^t} \\ \hat{v}_t &\leftarrow \frac{v_t}{1-(\beta_2)^t}. \end{aligned} \tag{8}$$

The decay parameters $\beta_1, \beta_2 \in [0, 1]$; and the model w_t^i is finally updated as follows:

$$w_{t+1}^i \leftarrow w_t^i - \alpha \frac{\hat{u}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{9}$$

where α is the upper bound of the update range of w_t, and ϵ is a small value to avoid zero division. In the Adam algorithm $\hat{u}_t/\sqrt{\hat{v}_t}$ represents the *signal-to-noise* ratio [28]. The ratio approximates to 1 when δ is stable, and the model parameters can be updated with a large learning rate close to the upper bound α. If δ fluctuates, the learning rate becomes smaller to detect the right direction carefully.

3.2.3 FedPGA algorithm: The details of the aggregation algorithm are presented in Algorithm 2 (see Fig. 4), and the hyper-parameters are adopted from the recommended parameters provided by [28]. In our design, each model starts from the same initial model weights w_0 and trains their own model in T times communication rounds. Each device conducts the training process in a parallel way. Note that, we only consider a synchronisation scheme for exchanging updates. At first, the device conducts the τ times local training using samples their own, and then calculates τ-Difference Gradient δ_t^i, and then exchange the δ_t^i in a partial way. After the device achieves the mixed gradient $\hat{\delta}_t^i$, it updates their local weight w_t^i using 'Adam' to obtain w_{t+1}^i.

Input: $N, T, \eta, \tau, w_0, \alpha, \beta_1, \beta_2, \epsilon, u_0, v_0$
1: **Each device** i **executes:**
2: initialize $w_0^i \leftarrow w_0$
3: **for** $t = 0, ..., T-1$ **do**
4: updates w_t^i for τ epochs of SGD on F_i with step size η to obtain $w_t^{i(\tau)}$.
5: calculate τ-Difference Gradient δ_t^i using Eq. (7)
6: split local δ_t^i into S slices.
7: randomly select S peers to pull partial τ-Difference Gradients.
8: merge the partial τ-Difference Gradient using Eq. (5) and Eq. (6) to obtain mixed τ-Difference Gradient $\hat{\delta}_t^i$.
9: update w_t^i using Eq. (8) and Eq. (9) to obtain w_{t+1}^i
10: **end for**

Fig. 4 *Algorithm 2: FedPGA*

3.3 Performance analysis

In this section, we will analyse the communication efficiency of FedPGA.

For each update i, the communication latency L^i consists of waiting time in the sending device and receiving device, denoted as L_{s}^i and L_{r}^r, respectively, and the transmission latency L_{trans}^i. We assume that the arrive rate of updates pulling requests at the device is λ^i according to a Poisson process. We denote the port bandwidth of sending device and receiving device is B_{s}^i and B_{r}^i, respectively. The data size of update is d. Thus, the data transfer time in the port of the sending device and receiving device is denoted as $\mu_{\mathrm{s}}^i = B_{\mathrm{s}}^i/d$ and $\mu_r^i = B_r^i/d$, respectively. The advantage of the *Queuing Theory*, we model the queuing and transfer process at the each port as M/D/1 model. Thus we have:

$$\begin{aligned} L_{\mathrm{s}}^i &= \frac{1}{\mu_{\mathrm{s}}^i} + \frac{\lambda^i}{2\mu_{\mathrm{s}}^i(\mu_{\mathrm{s}}^i - \lambda^i)} \\ &= \frac{d}{B_{\mathrm{s}}^i} + \frac{d\lambda^i}{2B_{\mathrm{s}}^i(B_{\mathrm{s}}^i - d\lambda^i)} \end{aligned} \tag{10}$$

$$\begin{aligned} L_{\mathrm{r}}^i &= \frac{1}{\mu_{\mathrm{r}}^i} + \frac{\lambda^i}{2\mu_{\mathrm{r}}^i(\mu_{\mathrm{r}}^i - \lambda^i)} \\ &= \frac{d}{B_{\mathrm{r}}^i} + \frac{d\lambda^i}{2B_{\mathrm{r}}^i(B_{\mathrm{r}}^i - d\lambda^i)} \end{aligned} \tag{11}$$

$$L_{\mathrm{trans}}^i = \frac{d}{B_{\mathrm{trans}}^i} \tag{12}$$

where the B_{trans}^i is the link bandwidth between the sending and receiving device. Notice that, we also assume that the $(\lambda^i/\mu_{\mathrm{r}}^i) < 1$ and $(\lambda^i/\mu_{\mathrm{s}}^i) < 1$.

Thus, combining (10), (12) and (11), the communication latency L^i for update i is:

$$L^i = L_{\mathrm{s}}^i + L_{\mathrm{r}}^i + L_{\mathrm{trans}}^i. \tag{13}$$

Finally, the latency of each device select S peer devices L is:

$$\begin{aligned} L &= L_{\mathrm{s}}^{j} + L_{\mathrm{r}}^{j} + L_{\mathrm{trans}}^{j} \\ &= \frac{d}{B_{\mathrm{s}}^{j}} + \frac{d\lambda^{j}}{2B_{\mathrm{s}}^{j}\left(B_{\mathrm{s}}^{j} - d\lambda^{j}\right)} \\ &+ \frac{d}{B_{\mathrm{r}}^{j}} + \frac{d\lambda^{j}}{2B_{\mathrm{r}}^{j}\left(B_{\mathrm{r}}^{j} - d\lambda^{j}\right)} + \frac{d}{B_{\mathrm{trans}}^{j}} \end{aligned} \tag{14}$$

where $j = \mathrm{argmax}\left(L^{i}\right), \quad i \in [S]$.

$$\begin{aligned} \gamma &= \frac{\alpha + d\lambda^{j}/2B_{\mathrm{s}}^{j}\left(B_{\mathrm{s}}^{j} - d\lambda^{j}\right) + d\lambda^{j}/2B_{\mathrm{r}}^{j}\left(B_{\mathrm{r}}^{j} - d\lambda^{j}\right)}{\alpha/S + (d/S)\lambda^{j}/2B_{\mathrm{s}}^{j}\left(B_{\mathrm{s}}^{j} - (d/S)\lambda^{j}\right) + (d/S)\lambda^{j}/2B_{\mathrm{r}}^{j}\left(B_{\mathrm{r}}^{j} - (d/S)\lambda^{j}\right)} \\ &\leq \frac{\alpha + d\lambda^{j}/2B_{\mathrm{s}}^{j}\left(B_{\mathrm{s}}^{j} - d\lambda^{j}\right) + d\lambda^{j}/2B_{\mathrm{r}}^{j}\left(B_{\mathrm{r}}^{j} - d\lambda^{j}\right)}{\alpha/S + 1/S\left(d\lambda^{j}/2B_{\mathrm{s}}^{j}\left(B_{\mathrm{s}}^{j} - d\lambda^{j}\right) + d\lambda^{j}/2B_{\mathrm{r}}^{j}\left(B_{\mathrm{r}}^{j} - d\lambda^{j}\right)\right)} \\ &\leq S \end{aligned} \tag{15}$$

where $\alpha = \left(d/B_{\mathrm{s}}^{j}\right) + \left(d/B_{\mathrm{r}}^{j}\right) + \left(d/B_{\mathrm{trans}}^{j}\right)$. Notice that, for the convergence performance, we prove the FedPGA through empirical experiments in Section 4.

4 Performance evaluation

4.1 Experiments setup

4.1.1 Datasets and models: We use datasets and models from **LEAF** [29], an open-source benchmarking framework for federated settings, including six tasks. We summarised the statistics of datasets in Table 1. Additional details on the models and datasets are presented as follows:

- *Federated extended MNIST (FEMNIST)*: We study an image classfication problem on EMNIST dataset [30], which has 62-class. The federated version of EMNIST, called FEMNIST, split the dataset into different workers, i.e. each worker has a corresponding writer of digits/characters in EMNIST. We create the FEMNIST dataset in **LEAF** by using command./`preprocess.sh -s iid –sf 0.05 -k 100 -t sample –tf 0.8`. The model used takes as input a 28×28 image, followed with two convolution layers and two dense layers, and the output is a class label between 0 and 61.
- *Synthetic:* We create a diverse set of synthetic datasets, with different task numbers, class numbers, and worker numbers. This dataset follows a similar set up in [1, 31]. The logistic regression model takes as input a 60 dimension feature. (i) Synthetic-C5-W40: We generate the whole dataset with 1000 tasks, and sample the dataset using command./`preprocess.sh -s iid –sf 1.0 -k 5 -t sample –tf 0.8 -iu 0.001`, to have a 5 prediction classes model and 40 workers dataset. (ii) Synthetic-C5-W80: We generate a 5 prediction classes model and 80 workers dataset.

Table 1 Statistics of datasets

Dataset	Workers	Param.	Samples/worker	
			Mean	Std
FEMNIST	35	26,414,840	1144.89	392.77
Synthetic-C5-W40	40	1220	2688.82	0.38
Synthetic-C5-W80	80	1220	1344.41	0.49

4.1.2 Experiment implementation details:

- *Hardware device:* We simulate the distributed federated learning (each device performs local training and aggregation) on a server with 2 Intel(R) Xeon(R) E5-2650 v4 @ 2.20 GHz CPUs and 4 Nvidia 1080Ti GPUs.
- *Libraries:* We implement all code in TensorFlow 1.14.0. The full details could be found at https://github.com/ginger0106/BACombo.
- *Hyperparameters:* We split each dataset randomly on each worker into 80% training set and 20% testing set. The learning rates for all the datasets are 0.004; the batch sizes for all the datasets are 10. For the FedPGA, the default number of slices S is 8, Each worker uses SGD as a local solver, and the default number of mibi-batch SGD τ is 16. The bandwidth capacity of each worker is set to 100 Mb/s. The link bandwidth (Mb/s) among workers are uniformly sampled from {0.2, 0.4, 0.8, …, 7.8, 8}.

4.1.3 Baselines: We compare FedPGA with several baselines distributed federated learning methods:

- *Gossip:* Gossip is a distributed version of FedAvg, i.e. each device act as an aggregation server and a local update worker at the same time. At each communication round, each device randomly samples S peer workers and pull the *complete model weights*, then conduct the weight averaging using updates transmitted and local updates as FedAvg.
- *GossipPGA:* GossipPGA is a special case of FedPGA. At each communication round, each device randomly samples S peer devices and pull S *complete τ-Difference Gradients*, then merges them using the weighted averaging method as Definition 2 and conducts the 'Adam-like' adaptive updating.

4.1.4 Metrics:

- *Time:* To evaluate the convergence speed, we measure the average time to achieve 75% accuracy of all devices, which consists of local training time, updates (model weights, gradients, partial gradients) synchronising time, and updates transmission time.
- *Accuracy:* We also measure the average test accuracy of all devices at each synchronising iteration.

4.2 Experiment result

We now present the empirical results for `FedPGA`, we first investigate the performance of convergence of our proposed approach and compare the end-to-end convergence speed with other baselines, and demonstrate the superior performance of FedPGA. Then we present the impact of hyper-parameters in FedPGA.

4.2.1 Convergence property: We first investigate the performance of convergence about FedPGA compared with baselines. We present the whole training process over time, as illustrated in Fig. 5, FedPGA shows a good convergence performance as the traditional methods, FedPGA will convergence at the same test accuracy (78, 83, and 84%, respectively) among all the datasets (FEMNSIT in Fig. 5*a*, Synthetic-C5-W40 in Fig. 5*b* and Synthetic-C5-W80 in Fig. 5*c*, respectively) At the same time, FedPGA exhibits an obvious speedup ($14\times$, $13\times$, and $13\times$, respectively, compared to Gossip) in the convergence among all the datasets (FEMNSIT, Synthetic-C5-W40, and Synthetic-C5-W80) as shown in Fig. 5*d*.

Compared with the Gossip and GossipPGA, we could find that the adaptive updating method could increase the convergence rate by ($1.7\times$, $1.5\times$, and $1.4\times$, respectively, compared to Gossip) among all the datasets (FEMNSIT, Synthetic-C5-W40, and Synthetic-C5-W80) while the partial exchange could reduce the communication latency significantly ($8.2\times$, $8.5\times$, and $9.3\times$, respectively, compared to GossipPGA). As we discussed in

CAAI Trans. Intell. Technol., 2020, Vol. 5, Iss. 3, pp. 230–236

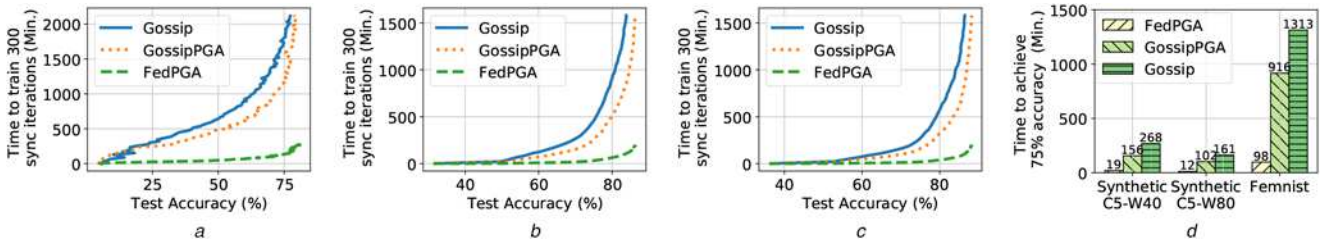

Fig. 5 *Convergence property*

a FEMNIST
b Synthetic-C5-W40
c Synthetic-C5-W80
d Convergence time

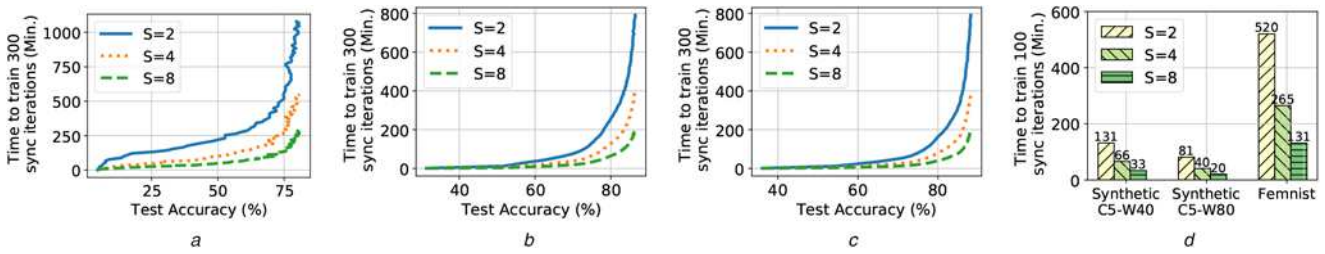

Fig. 6 *Impact of partial numbers*

a FEMNIST
b Synthetic-C5-W40
c Synthetic-C5-W80
d Convergence time

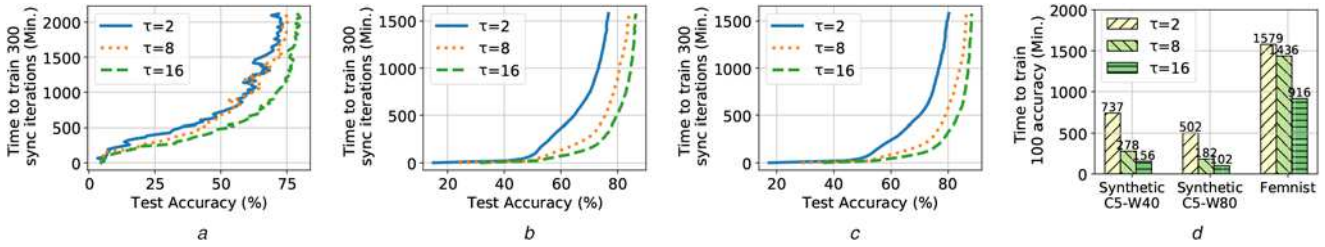

Fig. 7 *Impact of number of local training passes*

a FEMNIST
b Synthetic-C5-W40
c Synthetic-C5-W80
d Convergence time

Section 3, the theoretical speedup ratio in reducing communication bottleneck is less then the slice number, while the real end-to-end latency speedup is larger then slice number may be caused by the larger convergence rate aggregating more partial gradients.

4.2.2 Impact of partial numbers: The speedup of decentralised approaches comes from the removal of the bottleneck of the centralised server, and the advantage of FedPGA comes from the benefit of partial exchange. We measure convergence time with a different partial number of gradients S ($S \in \{2, 4, 8\}$) to investigate how it affects the training performance. Figs. 6*a–c* show that the accuracy of the partial results at each synchronisation iterations is not affected by the partition at all. Partitioning the model into eight slices ($S = 8$) has the same convergence trend as that without partition. While the synchronisation time is significantly reduced. As illustrated in Fig. 6*d*, by simply splitting the gradients into four slices can reduce the synchronisation time by half. This is because when $S = 4$, the original transmission quantity is divided into two parts and fed into $2\times$ more links. When the bandwidth is not exhausted, the sending and receiving time can be reduced almost proportionally.

4.2.3 Impact of number of local training passes: In this section, we investigate the impact of the number of local training passes. The τ is the number of local training passes. From Figs. 7*a–c*, we could observe that a larger number of local training passes could speed up the convergence, for instance, when $\tau = 16$, the time to achieve milestone accuracy is reduced by $1.7\times$, $4.7\times$, and $4.9\times$, respectively, compared to $\tau = 2$ among all the datasets (FEMNSIT, Synthetic-C5-W40, and Synthetic-C5-W80).

From Fig. 7*a*, we could observe that the accuracy shows a slight fluctuation when τ is small, especially in the bigger model. It is because the direction of τ-Difference Gradients is not stable when τ is too small, leading to the accuracy degradation in the training process.

4.2.4 Impact of the learning rate of adaptive updating: Since α in (9) sets magnitude of steps in parameter space in Adam, we investigate the learning rate α of adaptive updating as shown in Fig. 8. We could observe that different α

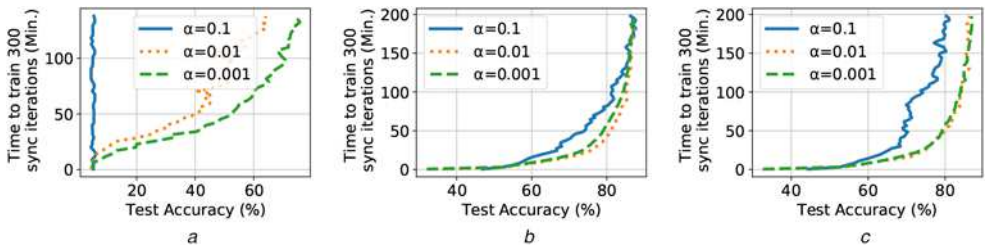

Fig. 8 *Impact of the learning rate of adaptive updating*

a FEMNIST
b Synthetic-C5-W40
c Synthetic-C5-W80

affect the convergence rate greatly. The larger α means a larger effective step-size, which leads to non-convergence or accuracy fluctuations. In Fig. 8*a*, when $\alpha = 0.1$, the model does not converge at all, and when $\alpha = 0.01$, it shows a great fluctuations in the accuracy curve. While the model of synthetic datasets is small and easy to train, the α affects the convergence slightly, especially when $\alpha = 0.01$ and $\alpha = 0.001$. However, when the number of participants becomes larger, a large step-size will bring more noise, resulting in worse convergence.

5 Conclusion

To avoid the drawback of network congestion in centralised parameter servers architecture in real-world federated learning scenarios, we explore the possibility of decentralised FL solution, called FedPGA. Taking the insight of philosophy in Adam, we design an adaptive model updating strategy. Our method also reduces the communication overhead by exchanging the partial gradient. The experiments show that FedPGA significantly reduces the training time and maintains a good convergence performance.

6 Acknowledgments

This work is funded by: National Key R&D Plan of China under Grant No. 2017YFA0604500, and by National Sci-Tech Support Plan of China under Grant No. 2014BAH02F00, and by National Natural Science Foundation of China under Grant No. 61701190, and by Youth Science Foundation of Jilin Province of China under Grant No. 20160520011JH & 20180520021JH, and by Youth Sci-Tech Innovation Leader and Team Project of Jilin Province of China under Grant No. 20170519017JH, and by Key Technology Innovation Cooperation Project of Government and University for the whole Industry Demonstration under Grant No. SXGJSF2017-4, and by Key scientific and technological R&D Plan of Jilin Province of China under Grant No. 20180201103GX, and by Project of Jilin Province Development and Reform Commission under Grant No. 2019FGWTZC001.

7 References

[1] McMahan, H.B., Moore, E., Ramage, D., *et al.*: 'Communication-efficient learning of deep networks from decentralized data'. Proc. of the 20th Int. Conf. on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282
[2] Mcmahan, H.B., Ramage, D., Talwar, K., *et al.*: 'Learning differentially private recurrent language models'. Int. Conf. on Learning Representations, Vancouver, BC, Canada, 2018
[3] Smith, V., Chiang, C., Sanjabi, M., *et al.*: 'Federated multi-task learning', 2017, pp. 4427–4437
[4] Hsieh, K., Harlap, A., Vijaykumar, N., *et al.*: 'Gaia: Geo-distributed machine learning approaching {LAN} speeds'. 14th {USENIX} Symp. on Networked Systems Design and Implementation ({NSDI} 17), Boston, MA, USA, 2017, pp. 629–647
[5] Zhang, B., Jin, X., Ratnasamy, S., *et al.*: 'Awstream: adaptive wide-area streaming analytics'. Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication, Virtual Event, Canada, 2018, pp. 236–252
[6] Daily, J.A., Vishnu, A., Siegel, C., *et al.*: 'Gossipgrad: scalable deep learning using gossip communication based asynchronous gradient descent', arXiv: Distributed, Parallel, and Cluster Computing, 2018
[7] Blot, M., Picard, D., Cord, M., *et al.*: 'Gossip training for deep learning', arXiv: Computer Vision and Pattern Recognition, 2016
[8] Patarasuk, P., Yuan, X.: 'Bandwidth optimal all-reduce algorithms for clusters of workstations', *J. Parallel Distrib. Comput.*, 2009, **69**, (2), pp. 117–124
[9] Vulimiri, A., Curino, C., Godfrey, B., *et al.*: 'Global analytics in the face of bandwidth and regulatory constraints', 2015, pp. 323–336
[10] Devlin, J., Chang, M., Lee, K., *et al.*: 'Bert: pre-training of deep bidirectional transformers for language understanding', arXiv: Computation and Language, 2018
[11] Watcharapichat, P., Morales, V.L., Fernandez, R., *et al.*: 'Ako: decentralised deep learning with partial gradient exchange', 2016, pp. 84–97
[12] Hu, C., Jiang, J., Wang, Z.: 'Decentralized federated learning: a segmented gossip approach', arXiv preprint arXiv: 190807782, 2019
[13] Nedić, A., Olshevsky, A.: 'Distributed optimization over time-varying directed graphs', *IEEE Trans. Autom. Control*, 2014, **60**, (3), pp. 601–615
[14] Jakovetić, D., Xavier, J., Moura, J.M.: 'Fast distributed gradient methods', *IEEE Trans. Autom. Control*, 2014, **59**, (5), pp. 1131–1146
[15] Shi, W., Ling, Q., Wu, G., *et al.*: 'A proximal gradient algorithm for decentralized composite optimization', *IEEE Trans. Signal Process.*, 2015, **63**, (22), pp. 6013–6023
[16] Boyd, S., Parikh, N., Chu, E., *et al.*: 'Distributed optimization and statistical learning via the alternating direction method of multipliers', *Found. Trends® in Mach. Learn.*, 2011, **3**, (1), pp. 1–122
[17] Koppel, A., Sadler, B.M., Ribeiro, A.: 'Proximity without consensus in online multiagent optimization', *IEEE Trans. Signal Process.*, 2017, **65**, (12), pp. 3062–3077
[18] Bedi, A.S., Koppel, A., Rajawat, K.: 'Asynchronous saddle point algorithm for stochastic optimization in heterogeneous networks', *IEEE Trans. Signal Process.*, 2019, **67**, (7), pp. 1742–1757
[19] Lin, T., Stich, S.U., Patel, K.K., *et al.*: 'Don't use large mini-batches, use local sgd', arXiv preprint arXiv: 180807217, 2018
[20] Wang, S., Tuor, T., Salonidis, T., *et al.*: 'Adaptive federated learning in resource constrained edge computing systems', *IEEE J. Sel. Areas Commun.*, 2019, **37**, (6), pp. 1205–1221
[21] Liu, Y., Xu, W., Wu, G., *et al.*: 'Communication-censored ADMM for decentralized consensus optimization', *IEEE Trans. Signal Process.*, 2019, **67**, (10), pp. 2565–2579
[22] Chen, T., Giannakis, G., Sun, T., *et al.*: 'Lag: lazily aggregated gradient for communication-efficient distributed learning'. Advances in Neural Information Processing Systems, Montreal, QC, Canada, 2018, pp. 5050–5060
[23] Sahu, A.K., Li, T., Sanjabi, M., *et al.*: 'Federated optimization for heterogeneous networks', arXiv: Learning, 2018
[24] Jiang, J., Hu, L., Hu, C., *et al.*: 'Bacomboâ€'bandwidth-aware decentralized federated learning', *Electronics. (Basel)*, 2020, **9**, (3), p. 440
[25] Leroy, D., Coucke, A., Lavril, T., *et al.*: 'Federated learning for keyword spotting'. ICASSP 2019–2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019, pp. 6341–6345
[26] Haddadpour, F., Kamani, M.M., Mahdavi, M., *et al.*: 'Local SGD with periodic averaging: tighter analysis and adaptive synchronization'. Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 2019, pp. 11080–11092
[27] Stich, S.U.: 'Local SGD converges fast and communicates little', arXiv preprint arXiv: 180509767, 2018
[28] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization', arXiv preprint arXiv: 14126980, 2014
[29] Caldas, S., Wu, P., Li, T., *et al.*: 'Leaf: a benchmark for federated settings', arXiv preprint arXiv: 181201097, 2018
[30] Cohen, G., Afshar, S., Tapson, J., *et al.*: 'EMNIST: extending MNIST to handwritten letters'. 2017 Int. Joint Conf. on Neural Networks (IJCNN), Anchorage, AK, USA, 2017, pp. 2921–2926
[31] Li, T., Sahu, A.K., Zaheer, M., *et al.*: 'Federated optimization in heterogeneous networks', arXiv preprint arXiv: 181206127, 2018