Efficient Ring-Topology Decentralized Federated Learning with Deep Generative Models for Medical Data in eHealthcare Systems

Zhao Wang 1,2,*, Yifan Hu 1, Shiyang Yan 3, Zhihao Wang 1, Ruijie Hou 1 and Chao Wu 1,

Abstract: By leveraging deep learning technologies, data-driven-based approaches have reached great success with the rapid increase of data generated for medical applications. However, security and privacy concerns are obstacles for data providers in many sensitive data-driven scenarios, such as rehabilitation and 24 h on-the-go healthcare services. Although many federated learning (FL) approaches have been proposed with DNNs for medical applications, these works still suffer from low usability of data due to data incompleteness, low quality, insufficient quantity, sensitivity, etc. Therefore, we propose a ring-topology-based decentralized federated learning (RDFL) scheme for deep generative models (DGM), where DGM is a promising solution for solving the aforementioned data usability issues. Our RDFL schemes provide communication efficiency and maintain training performance to boost DGMs in target tasks compared with existing FL works. A novel ring FL topology and a map-reduce-based synchronizing method are designed in the proposed RDFL to improve the decentralized FL performance and bandwidth utilization. In addition, an inter-planetary file system (IPFS) is introduced to further improve communication efficiency and FL security. Extensive experiments have been taken to demonstrate the superiority of RDFL with either independent and identically distributed (IID) datasets or non-independent and identically distributed (Non-IID) datasets.

Keywords: federated learning; medical data security; ring topology; deep generative model; privacy preserving; non-IID

1. Introduction

Recent years have witnessed a rapid growth of deep learning (DL) algorithms widely used to solve data-driven industrial problems in real-world medical applications [1,2]. These deep learning methods are benefited a lot by the massive amount of data collected. To improve the DL-based products, it brings great demand for different entities, e.g., huge amounts of devices belonging to different patients/hospitals, to contribute their data and train models together. In such collaborative training, the collection of massive data for centralized training causes serious privacy threats [3], which motivates federated learning (FL) [4] allowing participants to learn the model collaboratively by only synchronizing local-trained model parameters without revealing their original data.

A general federated learning system usually uses a central parameter server to coordinate the large federation of participating nodes (nodes, clients and workers are used interchangeably in this manuscript). For instance, the Conventional FL framework [5,6] uses a highly centralized architecture where a centralized node collects gradients or model parameters from data nodes to update the global model. Although some FL approaches...
have been proposed [7–10], the model training performance always suffers from low usability of medical data, such as data incompleteness, low quality, insufficient quantity and sensitivity. Deep generative models (DGM) like the generative adversarial network (GAN) can be used to tackle the problems mentioned above. In order to meet the data privacy constraints, distributed GAN algorithms are proposed [11]. Large communication bandwidth among nodes is required in current distributed GAN algorithms [12,13], while an intermediary is required to ensure convergence due to its architectures, which separate generators from discriminators. However, the communication bandwidth could be limited and costly in many real-world applications [14]. The center node in the current FL framework suffers from communication pressure and communication bandwidth bottleneck [15,16]. Communication-efficient distributed GAN is still an open problem, and we propose a framework that places local discriminators with local generators and synchronizes occasionally.

Additionally, the aforementioned centralized FL frameworks could bring security concerns and suffer the risk of single-point failure. Through the literature review, the decentralized FL framework [17–19] has been proposed. The decentralized FL framework removes the centralized node and synchronizes FL updates among the data nodes, then performs aggregation. However, it still faces challenges in communication pressure and cost, especially when blockchain is employed as an effective decentralized storage and replaces the central FL servers [18,20]. In addition, it is important to design the aggregation algorithm used in the decentralized FL framework that can achieve competitive performance under the situation of data poisoning from malicious nodes.

To tackle the aforementioned problems, a ring-topology decentralized federated learning (RDFL) framework is proposed in this paper. RDFL aims to provide communication-efficient learning across multiple data sources in a decentralized structure, which is also subject to privacy constraints. Inspired by the idea of ring-allreduce (https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/) which is accessed on 21 February 2017, consistent hashing technique [21] is employed in the proposed RDFL to construct a ring topology of decentralized nodes, which is able to reduce the communication pressure and improve topology stability. Besides, an innovative model synchronizing method is also designed in RDFL to benefit the bandwidth utilization and decentralized FL performance. Additionally, an InterPlanetary File System (IPFS) [22] based data sharing scheme is also designed to further improve communication efficiency and reduce communication costs. The code of RDFL will be published online soon.

To sum up, the main contributions of the proposed RDFL are as follows:

1. A new data node topology mechanism for decentralized FL has been designed in this work. The proposed mechanism is able to reduce communication pressure and significantly improve system stability. To the best of our knowledge, this is the first attempt to conduct a data node topology design for communication-efficient decentralized FL.
2. A novel ring decentralized federated learning (RDFL) synchronizing method is designed to improve bandwidth utilization and training stability.
3. To improve the communication performance and security of the decentralized FL framework, an IPFS-based data sharing scheme is designed to reduce system communication pressure and cost.

2. Related Work

This work relates to two literature, federated learning and distributed/federated GAN. Federated learning has emerged as a new paradigm in a distributed machine learning setup [4] and became widespread by Google’s blog post (https://ai.googleblog.com/2017/04/federated-learning-collaborative.html) that is accessed in 06 April 2017. It [4] proposes an FL process that collects locally calculated gradients and aggregates them at the central node. To help build FL tasks, some centralized FL frameworks have been proposed. Representatives of these frameworks are FATE (https://fate.fedai.org/), TensorFlow-Federated (TFF) (https://www.tensorflow.org/federated), PaddleFL (https://github.com/
To avoid the problems caused by the centralized FL framework, the research on the decentralized FL framework has attracted much attention. In [19], it has proposed a decentralized FL algorithm based on the Gossip algorithm and model segmentation. Local models are propagated over a peer-to-peer network through sum-weight gossip. Roy et al. have proposed a peer-to-peer decentralized FL algorithm. Lalitha et al. have explored a fully decentralized FL algorithm [23]. A blockchain-based decentralized FL framework is presented in [18]. To overcome the communication problem of the decentralized FL framework, current research focuses on researching novel communication compression or model compression techniques to reduce the communication pressure. For instance, Hu et al. utilize the gossip algorithm to improve bandwidth utilization and model segmentation to reduce communication pressure [19]. Amiri et al. [24] and Konečný el al. [25] propose model quantification methods to reduce communication pressure. Tang et al. [26] and Koloskova et al. [27] introduce communication compression methods to reduce communication pressure. Besides, sharing datasets [28] and knowledge distillation [29,30] are employed in FL to improve the FL performance on Non-IID datasets. To further protect the data privacy of FL, existing research focuses on several defense methods, including differential privacy [31] and multi-party secure computing (MPC) [32]. There are also reports on applying blockchain technology to decentralized FL to improve the security [7,18,33]. Distributed GANs have been proposed recently [34,35]. Moreover, they propose a single generator at the intermediary and distributed discriminators in [34]. A gossip approach for distributed GAN that does not require an intermediary server is presented in [35]. In order to deal with non-iid data, an individual discriminator is trained separately while the centralized generator is updated to fool the weakest discriminator in [11]. All of the above works require large communications during training. Few attempts have been made to address the problem of GAN training in an FL way [36–38], while little attention has been paid on improving communication efficiency.

3. The Proposed RDFL

In this section, we first describe how the designed topology mechanism in RDFL that utilizes a consistent hashing algorithm to build a ring decentralized FL topology for FL nodes. Then, we describe the synchronizing method in RDFL. Finally, an IPFS-based data sharing scheme is presented to further reduce communication cost.

3.1. Ring Decentralized FL Topology

**Topology Overview** Consider a group of \( n \) data nodes among which there are \( m \) trusted data nodes and \( n - m \) untrusted data nodes. These \( n \) data nodes are represented by the symbol \( D = \{DP_1, DP_2, DP_3, \ldots, DP_n\} \). RDFL utilizes a consistent hashing algorithm to construct a ring topology of \( n \) data nodes. The consistent hash value \( H_k = \text{Hash}(DP_k) \subseteq [0, 2^{32} - 1] \), \( DP_k \) represents the ip of \( DP_k \), \( k \in [1, n] \). Data nodes are distributed on the ring with an index value range \( [0, 2^{32} - 1] \) according to the consistent hash value. Figure 1 shows the ring topology constructed by the consistent hashing algorithm.

**Malicious Node** The malicious nodes can be detected with committee election methods [18]. The malicious nodes will only send local models to the nearest trusted data node found with the proposed ring topology in a clockwise direction and will not be passed anymore. In Figure 1, the green data nodes represent trusted data nodes and the gray data nodes represent Untrusty data nodes. According to the clockwise principle, Untrusty data nodes \( DP_2 \) and \( DP_3 \) send models to the trusted data provider \( DP_4 \). Untrusty data node \( DP_5 \) sends models to the nearest trusted data node \( DP_k \). With the help of a consistent hashing algorithm, different untrust data nodes can only send models to their corresponding trusty nodes, which reduces the communication pressure of trusty node effectively.
In order to deal with continuous untrusty nodes, a possible solution is to make the distribution of trusty nodes on the ring uniform. Hence, virtual nodes of trusted nodes can also be added to the ring topology if needed, which aims to further reduce communication pressure. Figure 2 shows a ring topology with virtual nodes. The green nodes with red dashed lines represent virtual nodes. $DP_1^{v1}$ is the virtual node of $DP_1$.

3.2. RDFL Training

**Synchronizing progress** Based on the ring decentralized topology constructed with the consistent hash algorithm, the trusty node follows the synchronizing progress illustrated in Figure 3. $M_i$ represents the model of data node $DP_i$. $r$ represents the number of rounds to execute model synchronization and $m$ represents the number of trusted nodes. At each synchronizing round, each node sends its models in a clockwise direction, then execute Federated Averaging (FedAvg) [4] to generate a new global model and starts the next iteration.

**Training progress with GAN Models** The horizon training iteration is denoted by $T$ and the index time is denoted by $t$. Consider nodes $DP_i, i \in 1, 2, \cdots, N$ with local dataset $R_i$ for each node, the weight of node $DP_i$ is denoted by $p_i$. Assume each node has local discriminator and generator with corresponding parameters $d^i$ and $g^i$, loss function $L_D$ and
$L_C^i$, local stochastic gradients $\tilde{\theta}_i(d^i_t, g^i_t)$ and $\tilde{h}_i(d^i_t, g^i_t)$ and learning rate $lr^d(t)$ and $lr^g(t)$ at time $t$. We assume the learning rates are the same across nodes. To improve the bandwidth utilization between trusted nodes, RDFL introduces the Ring-allreduce algorithm and the clockwise principle. As show in Figure 3, the trusted node $DP_1$ retains the local model $M_1$ after distillation, $DP_2$ retains $M_2$ and $DP_n$ retains $M_n$. Then, the trusted nodes utilize the Ring-allreduce algorithm and the clockwise rule to synchronize the local models of the trusted nodes. After synchronization, all trusted nodes have local models of other trusted nodes. The detail of RDFL training is described in Algorithm 1.

![Figure 3. Ring decentralized federated learning.](image)

**Algorithm 1** RDFL training with generative adversarial network.

**Input:** Set training period $T$, synchronizing interval $K$. Initialize global discriminator and generator $d_0$ and $g_0$. Initialize local discriminator and generator parameters $d^i_0 = d_0$, $g^i_0 = g_0$ for all $N$ nodes $DP_i, \forall i \in \{1, 2, \cdots, N\}$.

**Output:** The New global model

**Procedure:** Data Node Executes

1. for each FL round $t = 1, 2, 3, \ldots, T$ do
2. Each node calculates local stochastic gradient $\tilde{\theta}_i$ and $\tilde{h}_i$ corresponding to local discriminator and local generator respectively while fake generated by the local generator.
3. Each updates its local parameter in parallel;
   \[
   \begin{align*}
   d^i_t & \leftarrow d^i_{t-1} + lr^d(t)\tilde{\theta}^i_t \\
   g^i_t & \leftarrow g^i_{t-1} + lr^g(t)\tilde{h}^i_t 
   \end{align*}
   \]
4. if $t \mod K = 0$ then
5. Malicious node detection
6. Each trusted node receive all trusty nodes’ model parameters through the ring;
7. $B$ is the subset of $N$ stands for trusty nodes
8. for $DP_i \in B$ executes global model parameters
9. Each node updates its local model parameters with the executed global parameters;
10. end if
11. end for

It needs to be pointed out that we assume all nodes on the ring participate in the communication process. If part of the nodes meet communication failures during parameter sending, an extension work could be taken by following [25]. Due to the paragraph
limitation, the proof of convergence for model averaging would not be listed here, a similar proof could be referred in [36], which conducts a centralized FL framework with GAN.

3.3. IPFS-Based Data Sharing Scheme

In the most decentralized FL work [7,17–19,33,36], we noticed that the model parameters are transferred among data nodes directly, which occupies a lot of communication overhead and could cause serious communication cost when the blockchain is employed. For instance, gas fee is required in popular blockchain Ethereum, where the cost could be significantly high for large models. In order to reduce the risk of communication costs, an IPFS-based data sharing scheme is designed in RDFL. Data files, e.g., model parameters, in IPFS would be divided into multiple pieces stored on different nodes and IPFS will generate the IPFS hash corresponding to the file. The IPFS hash is a 46 byte string and the corresponding file can be obtained from IPFS through the IPFS hash.

As shown in Figure 4, data provider (node $DP_k$) sends its data, e.g., model parameters, to the data receiver (node $DP_h$):

1. Data provider creates an AES key.
2. Data provider stores data onto the IPFS and gets the corresponding IPFS hash.
3. Data provider encrypts the AES key in the above step using the public RSA key provided by the data receiver, which ensures that only the data receiver can conduct decryption to access the AES key.
4. **Data provider send the encrypted AES key to data receiver.**
5. **Data provider send encrypted IPFS hash to data receiver.**
6. Data receiver get the encrypt AES key and conducts decryption with its RSA private key.
7. Data receiver get the encrypted IPFS hash and conducts decryption with the received AES key in the above step.
8. Data receiver get the relevant file from IPFS with the IPFS hash.

The direct communication between data node $DP_k$ and $DP_h$ only occurs at Steps 4 and 5 in the proposed scheme, where the size of both the AES key and IPFS hash are significantly smaller than DGM or DNN model parameters. Therefore, the proposed IPFS data sharing scheme in RDFL is able to significantly benefit the system communication efficiently and reduce communication cost especially when the blockchain technique is used.

![Figure 4. Workflow of IPFS data sharing scheme.](image-url)
3.4. Communication and Computation Complexity

Since each node needs to train its local discriminator and generator, RDFL requires similar computations compared to FedGan [36] and increased computations (roughly doubled) for each node compared to distributed GAN [35]. The communications in the proposed RDFL are mainly limited to parameter transferring among all nodes in each round for every \( K \) steps. Assume \( M \) is the size of model parameters, including the discriminator and generator, there would be \( N - 1 \) times communications in one round and the average load per communication time per node is \( M \). Increasing \( K \) could reduce the communication frequency, which may reduce the performance of the FL train. An overview of another two decentralized FL communication methods is shown in Figure 5. A summary of communication complexity comparison is shown in Table 1. Generally, the total transferred data volume per FL round is similar for all three methods. The proposed RDFL achieves a better performance on the communication pressure of the nodes, which could benefit the system’s bandwidth utilization and increase robustness.

Table 1. Communication complexity analysis.

<table>
<thead>
<tr>
<th>Decentralized FL Framework</th>
<th>Communication Times/Round</th>
<th>Node Pressure (MB/c)</th>
<th>Total Transferred Data Volume per Round (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P</td>
<td>1</td>
<td>( N \times M )</td>
<td>( N^2 M )</td>
</tr>
<tr>
<td>FL Gossip [19]</td>
<td>( \text{round}\left(\frac{N-1}{2}\right) )</td>
<td>( 2M )</td>
<td>( 2NM \times \text{round}\left(\frac{N-1}{2}\right) )</td>
</tr>
<tr>
<td>RDFL</td>
<td>( N - 1 )</td>
<td>( M )</td>
<td>( N(N-1)M )</td>
</tr>
</tbody>
</table>

4. Experimental Results and Discussions

4.1. Experimental Setup

In this section, we conduct several experiments to evaluate the proposed RDFL to show its convergence, performance in generating close-to-real data, and robustness in reducing communications (by increasing synchronization interval \( K \)). The inception score (IS) is used in this paper, which is a common criterion in measuring the performance of GAN [12]. Another criterion used is the Earth Mover’s Distance (EMD), which is also known as the Wasserstein distance. In practice, EMD is approximated by comparing average softmax scores of drawn samples from real data against the generated data such that:

\[
\text{EMD}( (x_r, y_r), (x_g, y_g) ) = \frac{1}{N} \sum_{i=1}^{N} ( f_\theta(x_r^i) | y_r^i | - ( f_\theta(x_g^i) | y_g^i ) )
\]

where \((x_r, y_r)\) are real data samples, \((x_g, y_g)\) are generated data samples, \( f_\theta \) is the oracle classifier mentioned above. EMD measures a relative distance between real data and fake
data. Obviously, a better generator should have a lower EMD by producing realistic images closer to real images.

We build the training set of each client by randomly choosing 50% of the total training samples with replacements to simulate IID data. In order to further examine the performance of RDFL on a non-iid dataset, the latent dirichlet allocation (LDA) and the label partition method is applied to divide the dataset into $N$ partitions [39].

4.2. RDFL Training Performance with GAN

We test RDFL on MNIST to show its performance on image datasets. MNIST consists of 10 classes of data, which we split across $B = 5$. The hyperparameters of the GAN model used in the experiment are listed in Table 2.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Kernel</th>
<th>Strides</th>
<th>Feature Maps</th>
<th>BN</th>
<th>Non-Linearity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(z)100 \times 1 \times 1$ input</td>
<td>Trans Conv: $4 \times 4$</td>
<td>$1 \times 1$</td>
<td>256</td>
<td>Y</td>
<td>ReLU</td>
</tr>
<tr>
<td>Trans Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>128</td>
<td>Y</td>
<td>ReLU</td>
<td></td>
</tr>
<tr>
<td>Trans Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>64</td>
<td>Y</td>
<td>ReLU</td>
<td></td>
</tr>
<tr>
<td>Trans Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>3</td>
<td>N</td>
<td>Tanh</td>
<td></td>
</tr>
<tr>
<td>$D(x)32 \times 32 \times 3$ input</td>
<td>Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>32</td>
<td>Y</td>
<td>LeakyReLU</td>
</tr>
<tr>
<td>Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>64</td>
<td>Y</td>
<td>LeakyReLU</td>
<td></td>
</tr>
<tr>
<td>Conv: $4 \times 4$</td>
<td>$2 \times 2$</td>
<td>128</td>
<td>Y</td>
<td>LeakyReLU</td>
<td></td>
</tr>
<tr>
<td>Conv: $4 \times 4$</td>
<td>$1 \times 1$</td>
<td>1</td>
<td>N</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

From Figure 6, the trained GAN with RDFL is able to generate close-to-real images. We check Gan with RDFL performance robustness to reduced communications and increased synchronization intervals $K$ by setting $K = 1000; 2000; 5000; 10,000; 20,000$. The results are shown in the middle and right part of Figure 6, which indicate that Gan with RDFL has high performance for image data. Furthermore, its performance is robust to reducing the communications by increasing synchronization intervals $K$. In addition, we also conduct experiments for GAN with RDFL under the non-iid scenario. The results are shown in Figure 6. It could be seen that the GAN with RDFL could still finish training with acceptable image generation quality. We would like to encourage researchers to tackle the problem of federated learning of GANs with non-IID data in the future.
Figure 6. Illustration of FL training quality with GAN on MNIST IID (left) and non-IID (right), where number of nodes $B = 5$. (left) Generated images on $K = 2000$, (middle) IS vs. Iterations with $K \in [1000, 2000, 5000, 10,000, 20,000]$, (right) EMD vs. Iterations with $K \in [1000, 2000, 5000, 10,000, 20,000]$. 

5. Conclusions

In this paper, we propose a decentralized FL framework based on DGMs called RDFL to tackle the problems in existing decentralized FL frameworks. RDFL utilizes a consistent hashing algorithm and Ring-allreduce to improve communication performance, decentralized FL performance and stability. Moreover, RDFL introduces IPFS to further improve communication performance and reduce communication cost. We hope that RDFL can facilitate the application of decentralized FL with DGMs on medical areas. Future work will be focused on improving effective aggregation methods to replace the existing FedAvg algorithm. The related code will be published at https://github.com/ZJU-DistributedAI/RDFL-GAN accessed on 7 April 2021.

**Author Contributions:** Conceptualization, Z.W. (Zhao Wang), C.W. and Y.H.; methodology, Z.W. (Zhao Wang); software, Y.H.; validation, Y.H., R.H. and Z.W. (Zhihao Wang); formal analysis, R.H. and Z.W. (Zhihao Wang); investigation, Y.H.; resources, Y.H.; data curation, R.H. and Z.W. (Zhihao Wang); writing—original draft preparation, Z.W. (Zhao Wang); writing—review and editing, S.Y.; visualization, Y.H., R.H. and Z.W. (Zhihao Wang); supervision, C.W.; project administration, Z.W. (Zhao Wang); funding acquisition, C.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Key Research and Development Project of China (2021ZD0110400), National Natural Science Foundation of China (U19B2042) and Ningbo Natural Science Foundation (2021J167).
Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References


