This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Decentralized Gossip-Assisted Deep Learning Model Training for Resource-Constraint Edge Devices

Jatin Deep Singh, Neha Singh, *Graduate Student Member, IEEE*, Mainak Adhikari, *Senior Member, IEEE*, and Amit Kumar Singh, *Senior Member, IEEE*

***Abstract*—There is significant interest in edge computing (EC) for computational social systems to process and store data at the edge of the network. One of the key applications of EC is to analyze the large-scale social data, received from multiple sources using machine learning/deep learning (ML/DL) models with minimum delay and higher accuracy. However, traditional models are often large and require significant computational resources, posing a challenge in resource-constrained edge networks. Besides that, traditional centralized ML/DL methods including collaborative learning have limitations such as data privacy and communication overhead. Federated learning (FL) is an alternative solution to overcome some of these limitations by allowing model training across multiple decentralized devices without sharing the actual data. However, the standard FL approaches face some challenges, including extended training times due to the heterogeneous devices and the risk of single-point failure. To address these challenges, in this article, we propose a novel Gossip-assisted DL model for resource-constraint edge devices problem by enabling decentralized and serverless training while mitigating the risk of single-point failure. Besides that, we develop a lightweight model extractor for local edge devices to train a DL model with the collaboration of neighboring devices that improves knowledge discovery with higher prediction accuracy. Extensive simulation results over two publicly available large-scale datasets demonstrate the effectiveness of the proposed approach over the state-of-the-art techniques.**

***Index Terms*—Decentralized training, deep learning (DL), edge computing (EC), gossip learning (GL), knowledge discovery.**

I. Introduction

THE advent of edge computing (EC) in a distributed environment extends the capability of the cloud services at the edge of the network and analyses the data with minimum delay. Due to this, various social applications such as Azure IoT edge and AWSGreengrass to open source platforms such as Apache Edgent and KubeEdge adopt deep learning (DL) models to analyze large-scale data at the edge devices [1]. However, the current state of EC platforms is characterized by high heterogeneity with approximately two million types of device configurations [2]. This diversity arises from two main factors. First, EC platforms possess a variety of hardware options, including accelerators like TITAN Xp or RTX 2080 Ti Graphics Cards, and different CPU frequencies. Second, the allocation of resources on these platforms is distributed among both training tasks and concurrently executing workloads, often characterized by their transient nature and irregular inference outcomes.

In recent times, DL models have revolutionized multiple computational social applications including image recognition, natural language processing, and robotics. However, their training often demands substantial computational power and memory, which presents challenges for deployment on resource-constrained edge devices such as smartphones, microcontrollers, and drones. These edge devices typically feature limited processing power rendering traditional DL model training methods impractical. The development of distributed learning within a set of devices, connected in a network has introduced various methodologies such as data and model parallelism [3]. Other approaches prioritize the asynchronous training algorithm to address challenges, posed by unreliable networks [4]. Federated learning (FL), one of the potential distributed learning models, employs various approaches to enable large-scale dataset training across multiple participants (i.e., local edge devices) in the network. The majority of these strategies rely on a central device/server, which has the potential to become a system bottleneck. In the field of FL, several challenges have been identified that demand attention and resolution. Some of these challenges are listed as follows.

1) *Single-point failure*: In the traditional FL, a central server is typically responsible for coordinating the training process and aggregating model updates from participating devices/nodes and sending them back to their coordinating devices. This poses a risk, as the entire FL system can be compromised if the central server fails or becomes unavailable.

Manuscript received 26 February 2024; revised 10 April 2024, 19 April 2024, and 26 April 2024; accepted 28 April 2024. This work was supported by DST (SERB), Government of India, under Grant SRG/2022/000071D. *(Corresponding author: Mainak Adhikari.)*

Jatin Deep Singh and Neha Singh are with the Department of Computer Science, Indian Institute of Information Technology, Lucknow 226002, India (e-mail: mcs22006@iiitl.ac.in; neha08hs@gmail.com).

Mainak Adhikari is with the School of Data Science, Indian Institute of Science Education and Research, Thiruvananthapuram 695551, India (e-mail: mainak.ism@gmail.com).

Amit Kumar Singh is with the Department of Computer Science and Engineering, National Institute of Technology, Patna 800005, India (e-mail: amit.singh@nitp.ac.in).

Digital Object Identifier TCSS.2024.3395516

2329-924X © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

2) *Straggler effect*: In traditional synchronous FL aggregation (Fedavg), the global server keeps waiting for all coordinating devices to compute their computations before it can proceed with global aggregation. This process is hampered by the straggler problem, where the overall system is slowed down due to the limited computational capabilities of some edge devices.
3) *Data heterogeneity*: In FL, the presence of non-IID (non-identically independently distributed) data distributions poses a significant challenge when the data across participating devices exhibit varying statistical properties [5]. This divergence in data characteristics complicates the process of aggregating model updates from distributed edge devices, leading to performance degradation and compromised model quality.

To address these challenges, gossip-based learning has emerged as a promising solution to mitigate these limitations by leveraging decentralized communication protocols such as Wi-Fi. In Gossip learning (GL), decentralized communication among edge devices fosters knowledge discovery by enabling nodes within a network to exchange information and update model parameters directly with a subset of their neighbors. This approach facilitates collaborative learning, allowing nodes to leverage diverse datasets and experiences without relying on a central server for coordination. GL algorithms iteratively update model parameters based on local data and exchange information with neighboring edge devices in a randomized manner. However, traditional GL approaches may face challenges, primarily, stemming from its adherence to a ring topology and its utilization of random gossiping through the selection of a singular edge device as an aggregator. This approach limits connectivity to only neighboring edge devices, failing to foster full collaborative learning.

Motivated by the limitations of GL, we have developed an extended GL framework to overcome the existing challenges. In this new framework, we establish a fully connected graph topology where every edge device is directly connected to each other. This comprehensive connectivity ensures that all edge devices can effectively collaborate in the learning process. Within this framework, we designate one edge device as the source aggregator, responsible for aggregation tasks.

To ensure robustness and fault tolerance, we have integrated the Bully algorithm into the proposed framework. In the event of a failure of the current source aggregator, the Bully algorithm dynamically selects a new source aggregator to maintain the continuity of the learning process. During this transition period, periodic connections with the cloud are established to facilitate real-time data transfer for model updates. The cloud serves as a repository of knowledge, enabling seamless transitions and updates in case of a change in the source aggregator. Additionally, the cloud assists the new source aggregator in updating its aggregation knowledge, ensuring the integrity and accuracy of the learning process.

This extended GL framework offers a comprehensive solution that addresses the drawbacks of traditional GL methods. By leveraging full connectivity, dynamic source aggregation selection, and cloud integration, we enhance the efficiency, reliability, and scalability of collaborative learning in EC environments. The main contributions of the proposed approach are summarized as follows.

1) *Lightweight model extraction*: To establish a GL environment with edge devices, the first step entails designing a lightweight neural network model, capable of operating efficiently within the resource-constraint edge devices. Initially, the standard Convolution Neural Network (CNN) model is trained on the cloud, utilizing large datasets using a lightweight model extractor. Once developed, the lightweight model is deployed onto the edge devices for further refinement through local training.
2) *Novel GL framework*: We have designed a novel GL framework that leverages a fully connected graph, ensuring every edge device is interconnected. Within this framework, a designated source edge device assumes the role of an aggregator, facilitating full collaborative learning. In the event of a failure of this source aggregator, we implement the Bully algorithm to autonomously select a new source aggregator. Meanwhile, we periodically establish connections with the cloud to transmit real-time data for model updates. When the Bully algorithm designates a new source aggregator, the cloud server assists this device in updating the aggregation knowledge. Through this dynamic framework, we unlocked the potential for unparalleled collaboration and resilience in distributed environments.

To validate and enhance the effectiveness of the proposed GL framework utilizing a lightweight DL model, we conduct comprehensive experiments using two benchmark large-scale datasets. These experiments are meticulously designed to evaluate the performance of the developed systems under various conditions and scenarios, providing empirical evidence to substantiate the proposed framework and approach. The subsequent sections of this article are structured as follows: Section II delves into the related works. Section III offers a comprehensive discussion of the proposed system model and the problem formulation. The extraction of the lightweight model and Gossip architecture along with its learning approach are introduced in Section IV. Section V outlines the experimental setup and offers a detailed analysis of the observed results. Section VI summarizes the work.

II. Related Work

GL, an evolving paradigm in collaborative and decentralized machine learning (ML) has attracted considerable attention in recent studies. This section provides a summary of key developments and progress in the wider area of joint learning and methods based on Gossip communication in edge networks. Zhao et al. [6] proposed a novel decentralized FL framework that combines federated regularized nonlinear acceleration-based local training with a random broadcast gossip-based mechanism. This model reduces communication rounds and the complexity per iteration while preserving model performance. Barroso-Fernández et al. proposed a novel gossip-based

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

approach for distributed DL, aiming to alleviate communication bottlenecks in large-scale systems [7]. Satyanarayanan et al. [8] coined the term “cloudlets” to describe small, well-equipped data centers, located at the edge of the network. These cloudlets are intended to deliver robust computing power near mobile devices, essentially embodying the concept of EC. Shi et al. [9] discussed the role of EC in enabling real-time data analytics, highlighting how edge devices can process data locally using ML algorithms. Wu et al. [10] proposed a framework for object detection in low-light images using EC. This approach often entails sending sensitive data back to a central server, which can result in network overload due to the transmission of substantial amounts of data. To address these challenges, researchers in [11] have introduced FL, an approach where the model is trained collaboratively without sharing the actual data.

Significant contributions to address open problems in FL have been made by authors in [12] and [13], showcasing advancements of distributed learning in edge networks. Nonetheless, the reliance on a single server in this model could disrupt the training process due to potential server failures. Decentralized versions of FL have emerged as an important area of research, offering alternatives to the traditional centralized FL model [14]. Beltrán et al. [15] discussed decentralized DL that addresses the limitations of centralized FL by enabling decentralized model aggregation and minimizing reliance on a central entity. He et al. [16] proposed a hierarchical FL incentive mechanism in a UAV-assisted EC environment.

Han et al. [17] proposed the use of Gossip-based training of DL models on EC platforms, specifically addressing the challenges of heterogeneity and performance variation among edge devices. Zhou et al. [18] introduced lightweight ML models for edge devices to tackle the challenge of resource-constraints edge devices. Bhuyan and Chakraborty [19] have focused on data extraction across various domains, employing techniques such as particle swarm optimization, global and local searching, feature ranking, feature clustering, and multiobjective optimization [20]. Le et al. [21] have investigated the use of nonorthogonal multiple access assisted by intelligent reflecting surfaces to minimize the total latency by reducing the latency per training round. Additionally, multiple DL-assisted FL models have been developed including [22], [23], [24], and [25] for building effective models in edge networks.

The existing strategies resolve the issues of data heterogeneity and model performance in edge networks. However, the discussion lacks consideration for single-point failures and the need for lightweight models to facilitate faster convergence of DL models in edge devices for knowledge discovery, i.e., extracting valuable insights from the distributed data across the network. To address the challenges, in this article, we develop a new lightweight DL model for supporting model training and knowledge discovery on local edge devices and introduce a new GL for improving the model performance by sharing model parameters locally instead of sending them to a centralized cloud server. This can reduce the straggler effect of the edge devices and handle heterogeneous data efficiently.

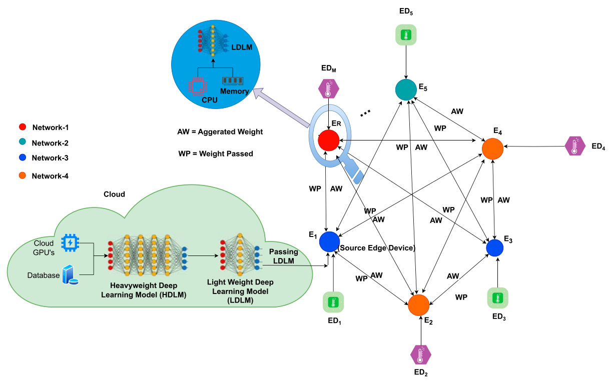

Fig. 1. Proposed Gossip-assisted edge networks.

III. System Model and Problem Formulation

In this section, we initially discuss the system model of the proposed GL-assisted DL framework. Subsequently, we present the problem statement, accompanied by an empirical observation about the challenges encountered in the standard FL scenarios during data analytics.

A. System Model

The proposed Gossip framework, presented in Fig. 1 consists of a set of distributed end devices like sensors, denoted as $ED = \{ED_1, ED_2, ED_3, ED_4, \dots, ED_M\}$, which transmit the data to the resource-constraint edge devices, represented as $E = \{E_1, E_2, E_3, E_4, \dots, E_R\}$, and a resource-rich centralized cloud server, connected within a network. The color-coded in Fig. 1 signifies the interconnection of edge devices across the networks, fostering interactive Gossip communication among them. In the proposed framework, initially, the centralized cloud server trains a DL model over a publicly available large-scale dataset using a heavyweight deep learning model (HDLM) to extract a lightweight model that is further deployed on edge devices.

Large-scale dataset residing in the cloud database is represented by $D = (q_r, s_r)$, where q_r and s_r denote the features and labels of the dataset, respectively. A subset of the original large-scale dataset, denoted as $D^{'} = (p_r, l_r)$, contains elements $p_r \in q_r$ and $l_r \in s_r$ utilized to construct the HDLM. Due to the resource-constraint nature of the local edge devices, the HDLM model is unable to train data and make a decision from those devices. Thus, one of the main requirements of the proposed framework is to generate a lightweight deep learning model (LDLM) from the trained HDLM that supports data analytics in the resource-constraint devices. The conversion of the LDLM is carried out in the stable public cloud server, enabling a smooth transition of LDLM to the edge device. Among the set of local edge devices, one edge device, namely the source edge device acts as a coordinator between the centralized cloud server and the local edge devices. The main purpose of the source edge devices is to deploy the initial LDLM model into the local edge devices and asynchronously aggregate the model parameters, received from the participating edge devices for improving the performance of models for knowledge discovery.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Here, we select one edge device as an aggregator that causes the issue of single-point failure of the source edge device. To handle this, we introduce a standard Bully algorithm for handling fault tolerance mechanisms. Thus, if the initially selected source edge device is damaged or unable to respond, then using the logic of the standard Bully algorithm another edge device from the network takes the responsibility of the source edge device and establishes coordination between a cloud server and local devices. The local edge devices further train the received LDLM model with their local dataset for knowledge discovery. Gossip communication between edge devices is established when one or multiple edge devices fail to reach the accuracy level of the model while training over the local datasets. In such a scenario, the edge device coordinates with its neighboring edge devices to improve the model performance. The neighboring edge devices transmit the model parameters to the source edge device.

The source edge device aggregates the model parameters using the proposed logic and upgrades the model of the requesting edge device for further analysis. Thus, using the proposed Gossip framework, the local model upgrades the performance with the help of neighboring model parameters instead of continuous communication to the centralized cloud server. This reduces communication overhead and congestion of the network while handling the straggler effect of slow-performing edge devices.

Algorithm 1: GL: Lightweight Model Extraction

```
Input: Neural Network Model: $r$, Threshold Model Accuracy: $\theta$,
       Threshold Model Size: $S_t$
Output: Lightweight Neural Network Model $L \in r$
Data: Dataset $D'$
Cloud Server Side:
$Train = D'[:, 0 : -1].values$
$Test = D'[:, -1].values$
$trained\_model = model.fit(Train, Test)$
Lightweight Model
$light\_model$ = None
for each $model\_k$ in $best\_model$ do
    $quantize\_model$ = Quantization($model_k$)
    $pruned\_model$ = Pruning($quantize\_model$)
    $size$ = $pruned\_model.size$
    if $size <= S_t$ then
        $light\_model \longleftarrow pruned\_model$;
return $light\_model$;
Best Lightweight Model
Initialize: $model[], best\_model[], count = 0$
for r=1 to R do
    model[r] = $trained\_Model(model(r))$
    $\alpha$ = model[r].accuracy if $\alpha > \theta$ then
        $best\_model[r] = model[r]$
        $count+ = 1$
return $LightweightModel(best\_model[r], S_t)$
```

B. Problem Formulation

The main objective of the work is to introduce asynchronous model aggregation for efficient training and knowledge discovery of subordinate edge devices. Consider a network, comprising R edge devices, organized in a Gossip configuration. The source edge device, denoted as E_1, is linked to its neighbor E_2, which, further connects to E_3 and so forth until it reaches edge device E_R, connected back to E_1. Each edge device, labeled as r, where $r \in R$, possesses its local dataset $D_r^{''}$, used for training in subsequent iterations. The large-scale dataset is denoted as $D = \bigcup_{r=1}^{R} D_r^{''}$. Global learning aims to minimize loss on each edge device, achieving a comparable loss reduction rate across all edge devices (ω) through weight aggregation. A loss function, denoted as $\mathcal{L}(\omega)$ is formulated during the collective training of each edge device with samples from D. Training occurs concurrently on the edge devices, and the averaging method proves effective in reducing the loss function, defined as follows:

$$\text{minimize} \quad \mathcal{L}(\omega) = \frac{1}{|D|} \sum_{r=1}^{R} |D_r''||\mathcal{L}_r(\omega)|. \tag{1}$$

Here, $\mathcal{L}_r(\omega)$ represents the loss function of the rth edge device and $\mathcal{L}(\omega)$ is the global objective function, which is the weighted average of local objective functions $\mathcal{L}_r(\omega)$. The loss function is mentioned in (1) is solved using the proposed method, mentioned in Section IV.

IV. Proposed Gossip Learning Strategy

In the proposed GL strategy, the utilization of a lightweight model and asynchronous communication among edge devices is employed to aggregate weights at the source edge device. Subsequently, these aggregated weights are passed on to edge devices for additional retraining. The entire process is segmented into three main stages. Initially, an HDLM is developed and converted into an LDLM for edge devices. The second stage involves setting up a Gossip-based environment, leading to the final stage where Gossip communication occurs, enabling the collaborative training of the model.

A. Lightweight Neural Network Extractor

To address challenges associated with straggler edge devices, we have created a lightweight neural network extractor, aimed to obtain the most efficient model. The objective of this extractor is to accelerate the training time of models, deployed on edge devices in successive iterations, promoting faster convergence and reduced inference time. This optimization ensures that these models are well suited for deployment on resource-constrained devices, taking into account factors such as efficiency, resource utilization, and real-world constraints. The procedural approach for constructing the lightweight model extractor is suggested in Algorithm 1.

Initially, there are R edge devices, utilizing the various DL models on dataset D'. Techniques such as dropout and batch normalization are employed to enhance model performance. The proposed methodology involves training multiple neural network models encompassing both artificial neural network (ANN) and CNN architectures. The trained models (trained_model) are denoted as $M_1, M_2, ..., M_R$ in Fig. 2. effectiveness of various models is evaluated based on a predefined threshold model accuracy (θ), aiding in the extraction of

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

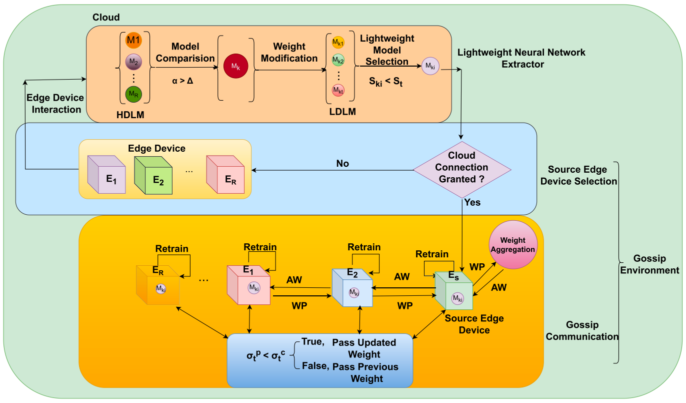

Fig. 2. Proposed Gossip-assisted DL model training.

models that exhibit superior performance (lines 2–4). Hence, out of the r models generated at various edge devices, a total of k models are selected for further analysis. Further, the selected models are optimized to be lightweight using various model optimization techniques, primarily focusing on quantization and pruning, to enhance the retraining phase (lines 7–12). Additionally, we compare the sizes of different lightweight models $(M_{k1}, M_{k2}, .., M_{kt})$ using the threshold model Size (S_t) to determine the best lightweight model, as outlined in lines 15–20. This is crucial for resource-constrained edge devices to improve storage utilization and speed up computations. We return the best lightweight model, which is deployed on edge devices (line 21).

B. Gossip Environment

The Gossip Environment represents the configuration used to replicate GL among different edge devices within a network. Algorithm 2 outlines the various activities that occur in the environment to facilitate Gossip interactions among the different edge devices. The setup for the Gossip environment involves two stages, described as follows.

1) Source Edge Device Selection: Initially, all edge devices interact with the cloud, based on their connection setup. The first edge device that successfully establishes a connection with the cloud, represented in lines 5–9, becomes the source edge device for the network. This prompts the other edge devices to cease their interaction with the cloud. If no initial source edge device is selected, the configurations of the edge devices are adjusted. Subsequently, connection requests from the edge devices are sent to the cloud again to establish a source edge device for the network (lines 10–11). In the later stage of the training process, if any additional edge devices wish to join the network, they can interact either with the source edge device or with any of the remaining connected edge devices in the network. In other scenarios, if the source edge device fails, the standard Bully algorithm is executed to select a new source edge device for the network. The source edge device interacts with the cloud once to get the updated best lightweight model for further learning to be continued, and it passes the aggregated weights later for self and other edge devices to enhance its performance (lines 12–13).

Algorithm 2: GL: Gossip Environment Setup

```
   Input: Edge Device: $R$, Cloud Services: $Cloud$, List of Edge
          Device: $E[R]$, Neural Network Model: $R$, Threshold
          Model Accuracy: $\theta$, Threshold Model Size: $S_t$
   Output: Trained Network of $R$ Edge Devices
   Data: Large-scale Dataset $D$
1  Edge Device Side:
2  Initialize: $source = None, train\_device = list()$
3  $source\_device$ = None, $temp = False$
4  for each $device\ in\ Edge\_Devices$ do
5      if $Cloud.connection\_granted(device)$ then
6          $device.is\_source = True$
7          $source\_device = device$
8          $temp = True$
9          return $source\_device$;
10 if $temp$ then
11     return $Source\_Edge\_Device(Cloud, Edge\_Devices)$;
12 for each $device\ in\ Edge\_Device$ do
13     Add $device$ to $network$ initially at $source$ and later to
         $source$ or other $device$;
14 $network.source\_device = source$
15 $source = Source\_Edge\_Device(Cloud, E)$
16 $source.model$ = Lightweight Model Extraction$(R, \theta, S_t)$
17 $network = Network\_Connection(E, source)$
18 return $newtork$;
```

The source edge device aggregates the weights of model parameters of the participating edge devices and transmits them to requesting edge devices to improve their performance from the next iteration. From their respective neighbors, the same aggregated weights are passed to their respective neighbors if they have not received the updated weights. The source edge device also uses the aggregated weights to retrain itself, indicating the creation of a Gossip network by adding the edge devices among themselves and assigning the source as source_device of the Gossip network (lines 16–19).

2) Initial Edge Device Training: The initial training of an edge device is structured to pave the way for Gossip to influence its role during the subsequent training phase. The device connected to the source edge device possesses a duplicate of the source edge device model. To initiate the training of edge devices, we employ transfer learning, combining feature extraction and fine-tuning elements. During the initial model training, each device trains on its local dataset, which is not visible to other edge devices in the network. The weight updation of the models is performed using

$$w_i(r) = w_i(r-1) - \eta\nabla(\mathcal{L}(w_i(r-1))) \quad (2)$$

outlining the weight update process, where the w_i indicate the weight, subscript i indicates the edge device index, r signifies the update step, η denotes the gradient step size, and ∇ being gradient of loss $\mathcal{L}$ with respect to previous iteration weight $w_i(r-1)$.

Now the Gossip network is established, which suggests the interaction of upcoming edge devices with the source and remaining connected edge devices in the network for Gossip communication. The purpose of this training phase is for edge

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 3: GL: Operations

```
Input: Source: $source$, Network: $network$,
Threshold_Time: $T$
Output: Trained Edge Device
1  $average\_weight = None$
2  $temp = network.source.list\_weight$
3  $length = len(temp)$
4  for each $item$ in $temp$ do
5      $average\_weight = average\_weight + item$
6  $average\_weight = average\_weight/length$
7  return $average\_weight$
8  for each $device$ in $network$ do
9      $device.aggregated\_weight = average\_weight$
10 $temp = copy(device.model)$
11 $\sigma_t^p = device.model.accuracy$
12 $temp.set\_weight(device.model.aggregated\_weight)$
13 $temp.model = Train\_Model(temp.model)$
14 if $temp.model.running\_time < T$ then
15     $\sigma_t^c = temp.model.accurcay$
16     if $\sigma_t^c > \sigma_t^p$ then
17         $device.model = temp$
             $device.pre\_weights = temp.get\_weights()$
             $w\_list.apppend(device.pre\_weights)$
18     return $device$
```

devices to comprehensively understand the data through the deployed model, facilitating further improvements.

C. Gossip Operations and Communication

The Gossip environment creates an atmosphere around the local edge devices, connected in the network to participate in the Gossip communication with the help of various logical steps, listed as follows.

1) Operations on Gossip Network: Algorithm 3 represents the operations involved in Gossip communication. The weights of initially trained network edge devices are propagated to the source edge device, where weight aggregation is performed by considering the weights of each layer and then averaging them across all edge devices, as shown in Algorithm 3 (lines 1–5). Consider two edge devices with weights across z layers, represented as $[[A'_1, A'_2, \ldots, A'_a], [B'_1, B'_2, \ldots, B'_b], \ldots [Z'_1, Z'_2 \ldots, Z'_z]]$, $[[A''_1, A''_2, \ldots, A''_a], [B''_1, B''_2, \ldots, B''_b], \ldots, [Z''_1, Z''_2 \ldots, Z''_z]]$.

The aggregated weight is expressed as follows:

$$AW = \left[\left[\frac{A'_i + A''_i}{2}\right]_{i=1}^{a}, \left[\frac{B'_i + B''_i}{2}\right]_{i=1}^{b}, \ldots, \left[\frac{Z'_i + Z''_i}{2}\right]_{i=1}^{z}\right]. \tag{3}$$

Once the aggregation is done, the aggregated weights are distributed to all other edge devices in the network (lines 8–9). If a device is slow or unable to transmit weights during a certain threshold time T, it is unable to contribute initially and acts as a slow edge device for successive iterations for further improvement. However, subsequent iterations aid the device in improving its performance. Further training of edge devices

Algorithm 4: GL: Communication

```
Input: Network: $network$, Accuracy_Change: $\delta$,
Rounds: $Iteration$ Gossip_Operation: $Operation$
Output: Enhance Gossip Network
1  Edge Device Side:
2  for $i = 1$ to $Iteration$ do
3      if $network.source$ is active then
4          $network.source.list\_weight = list()$
5          $temp = list()$
6          if $i == 1$ then
7              for $device$ in $network$ do
8                  $previous\_accuracy.append($
9                  $device.model.accuracy())$
10             $average\_weight =$
                 $Operation.Weight\_Aggregation(network)$
11         else
12             $accuracy = list()$
13             for $device$ in $network$ running parllely do
14                 $Retrain\_Device$
15                 $(temp\_device, threshold\_time)$
16                 $accuracy.append(temp\_model.accuracy)$
17             $current\_accuracy = accuracy$
18             $th = |(current\_accuracy - previous\_accuracy)|$
19             if $Any(th) > \delta$ then
20                 $previous\_accuracy = current\_accuracy$
21                 continue;
22     else
23         $network.release(source)$
24         $network.update\_connection()$
25         $network.source = Bully\_Algorithm(network)$
26 return $network$;
```

is executed to monitor the performance of each edge device. The previous model accuracy, denoted as σ_t^p, is compared with the current model accuracy σ_t^c across different iterations to update the model's previous_weights that is passed to other devices during the next iteration of GL for weight aggregation (lines 10–18).

2) Communication in Gossip Network: Algorithm 4 suggests the flow of GL across diverse edge devices within the Gossip network. The algorithm starts with the first iteration and checks for the source active devices. If the source devices are active, start the initial training for all the edge devices in the network and generate their model parameters with accuracy (lines 2–9). Now these model parameters are sent to the source edge device and aggregation is performed (line 10). In the successive iteration, all the edge devices receive the aggregated model parameters, start retraining the model, and save their current accuracy (lines 13–16). The difference in the previous accuracy and the current accuracy is stored in the th, which is then compared with the change in accuracy δ to identify the entire convergence of the network. δ is set by experimenting with different changes in accuracy over the various iterations.

Now, in successive iterations, all the identified slow edge devices send their updated model parameters to the source edge device along with their neighbors, just by keeping track of the edge device that does not contribute to the weight aggregation

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

in previous iterations (lines 17–22). This can reduce the communication overhead. To address the non-IID problem, each edge device is retrained on the limited instance of data that is trained on a large volume of incoming data to attain the previous and current knowledge on a variety of similar feature inputs. It helps the model to understand the variation in the data. In further iterations, the existence of the source edge device is checked by the network connections and if it fails then the Bully algorithm immediately activates and selects the temporary source edge device for the network (lines 24–26). In this whole process, periodically we establish the connection of the source edge device to the cloud that updates all the information related to aggregation, dataset, and model updation and select a new source edge device while the failure of the initial source edge device.

V. Experimental Analysis

This section describes the experimental setup of the proposed Gossip-assisted DL model training followed by the performance and comparative analysis of the proposed strategy.

A. Experimental Setup

The simulation is executed on a Dell Vostro workstation, equipped with a 12th Gen Intel® Core™ i7-12700 CPU. We developed a GL simulation setup in Python's virtual environment, integrated it into the development environment, and implemented all models using TensorFlow, creating client instances with TensorFlow objects on Kaggle for collaborative model training simulation. Leveraging the cloud computing capabilities of Kaggle, we instantiate 10 to 20 client objects. After conducting multiple experiments with various learning rates, we fix the learning rate η to 0.001, yielding optimal performance. The epoch set at 100 based on the designed callbacks in Keras, on validation_loss close to 0.0001 with no restoring of best_weight by model. Later a new upcoming edge device can connect to the source edge device and copy the same model architecture along with its previous_weight as none N and aggregated_weight from the connected edge device aggregated_weight. In each iteration, average accuracy and loss on each edge device are recorded, repeating the procedure multiple times.

B. Datasets

In the analysis, we opt to utilize the MNIST (Modified National Institute of Standards and Technology)[1] and CIFAR-10 (Canadian Institute for Advanced Research) datasets[2] due to their widespread use as benchmarks in ML research. MNIST comprises a collection of 70 000 handwritten digits (0–9), presented as grayscale images of size 28×28 pixels. CIFAR, on the other hand, offers two variants: CIFAR-10 and CIFAR-100, indicating the number of distinct classes present in each dataset (10 and 100, respectively). We split datasets into subsets of different sizes and distributed them across edge devices to mimic real-world scenarios. This helps explore how data diversity affects model performance and convergence in decentralized learning.

[1]https://www.kaggle.com/datasets/oddrationale/mnist-in-csv

[2]https://www.kaggle.com/datasets/fedesoriano/cifar10-python-in-csv

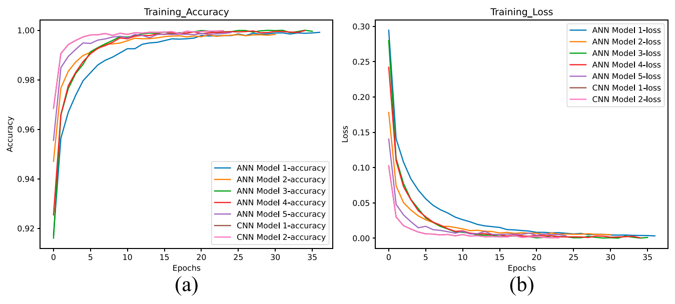

Fig. 3. Accuracy and loss analysis for model selection. (a) Accuracy. (b) Loss.

TABLE I
Model Performance Analysis

Model	Size (MB)	Train Accuracy (%)	Test Accuracy (%)
CNN-2	2.75	90.00	99.60
Quantize CNN-2	0.23	99.00	99.80
Pruned CNN-2	1.81	99.5	99.73

C. Performance Analysis

1) Lightweight Model Analysis: The extraction of the model is done on the comparative analysis made out of different ANN and CNN architecture on the label $(0-5)$ and finding the best possible model. Initially, we experimented with ANNs, comparing configurations with single and double hidden layers and varying activation functions between sigmoid and ReLU across models labeled 1–5. ReLU activation in hidden layers and softmax activation for output yielded superior performance. Extending this to CNNs, we began with CNN model-1 consisting of two convolutional layers and one dense layer, achieving a loss value of 0.0174. Subsequently in CNN model-2, we expanded the complexity by incorporating four hidden layers alongside the two convolutional layers, resulting in a notably reduced loss value of 0.0075 as shown in Fig. 3. Despite observing analogous accuracy across all models, the disparity in training loss compelled the extraction of the CNN model-2 (CNN-2).

The selected CNN-2 is used for the process of the lightweight building phase where the combination of quantization and pruning, and for the proposed strategy evaluation, quantization fits the best as shown in Table I. Hence, we opted for quantization over pruning. Figs. 4 and 5 illustrate the analysis of non-IID data distribution, and Figs. 6 and 7 illustrate IID data distributions for different state-of-the-art techniques and proposed strategies over different datasets. In the setup, different edge devices act for IID data distribution, the test split remains consistent across all edge devices. Conversely, for non-IID distribution, the test split across different devices is kept unequal, providing a maximum test split across different devices. After setting up the edge devices, they undergo training, and the analysis is conducted during training phase 1. Gossip iterations are performed on different models based on weight aggregation with some

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

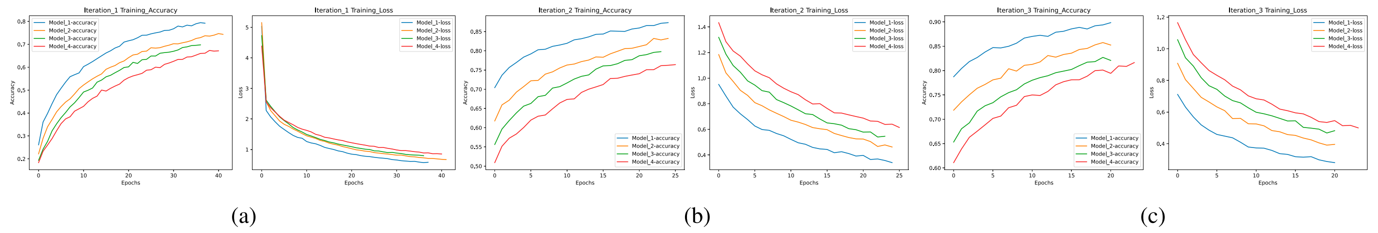

Fig. 4. Non-IID distribution for CIFAR-10 dataset using four different CNN models. (a) Training phase-1. (b) Training phase-2. (c) Training phase-3.

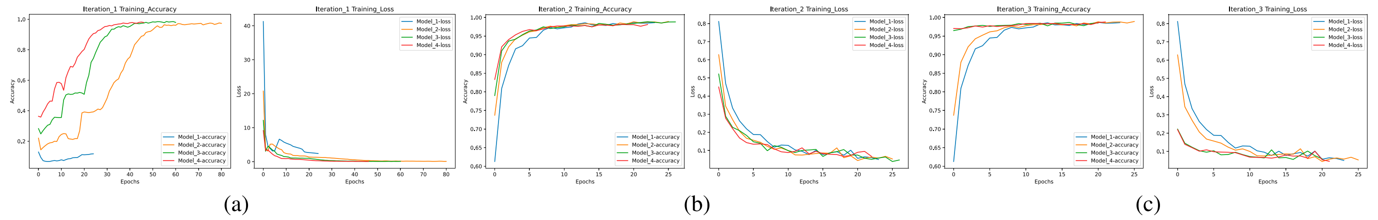

Fig. 5. Non-IID distribution for MNIST dataset using four different CNN models. (a) Training phase-1. (b) Training phase-2. (c) Training phase-3.

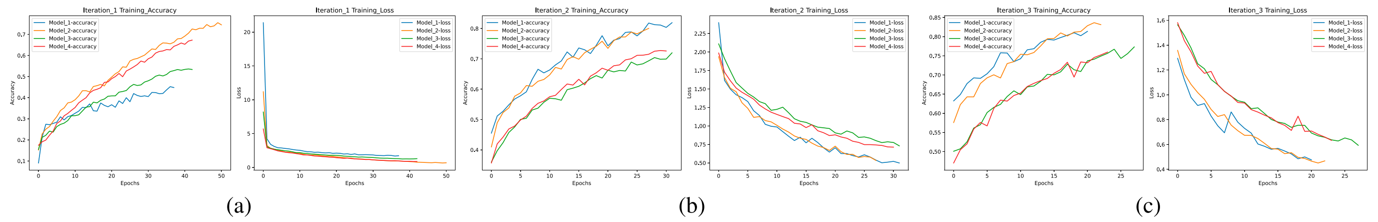

Fig. 6. IID distribution for CIFAR-10 dataset using four different CNN models. (a) Training phase-1. (b) Training phase-2. (c) Training phase-3.

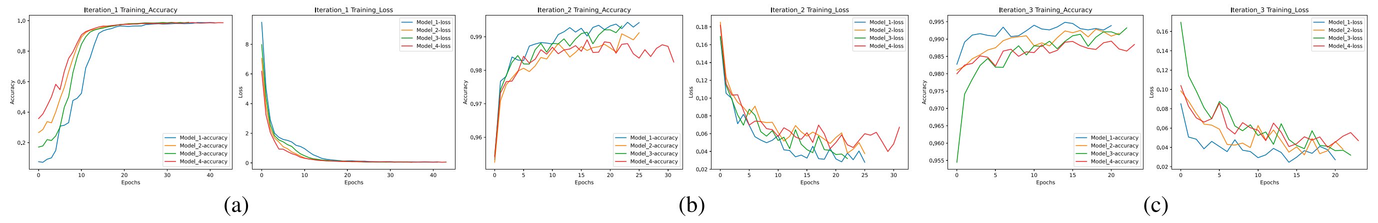

Fig. 7. IID distribution for MNIST dataset using four different CNN models. (a) Training phase-1. (b) Training phase-2. (c) Training phase-3.

edge devices being enhanced in further rounds and others being optimized in subsequent rounds.

In the initial training phase (phase 1), it is observed that different edge devices receive varying amounts of training data, resulting in discrepancies in accuracy. Specifically, node-1 exhibits lower accuracy compared to other edge devices (node-2, node-3, and node-4). However, through successive iterations of GL, all edge devices improve their accuracy and contribute to enhancing the performance of others. This phenomenon demonstrates the effectiveness of GL with convergence achieved around the 25th epoch. Detailed information regarding training time and test accuracy across the four different edge devices is provided in Tables II and III, respectively. The tables illustrate that the impact of stragglers is significantly reduced, and the performance of edge devices improves consistently throughout successive iterations.

TABLE II
EDGE DEVICE VERSUS ITERATION TEST ACCURACY (MNIST)

Data	I / E	Initial	1	2	3	4	5
IID	1	97.32%	97.82%	97.86%	97.90%	97.96%	98.10%
	2	97.96%	97.66%	98.10%	98.22%	98.48%	98.48%
	3	97.85%	98.19%	98.19%	98.22%	98.24%	98.24%
	4	97.77%	98.12%	98.39%	98.42%	98.42%	98.47%
Non-IID	1	50.63%	96.28%	96.28%	96.61%	97.84%	97.84%
	2	96.40%	97.48%	97.48%	97.53%	97.53%	97.86%
	3	97.13%	97.51%	97.84%	97.89%	97.89%	97.89%
	4	97.28%	97.69%	98.06%	98.06%	98.06%	98.30%

2) GL Analysis: The lightweight model is initially deployed on all edge devices to distinguish between them, as each edge

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
EDGE DEVICE VERSUS ITERATION TRAINING TIME (IN SECONDS) (MNIST)

Data	E \ I	Initial	1	2	3	4	5
IID	1	310.20	153.20	130.50	150.65	92.45	82.45
	2	400.05	250.25	150.30	142.38	81.25	110.35
	3	300.05	175.50	150.30	138.35	99.30	100.40
	4	350.15	160.45	175.45	160.45	120.28	120.38
Non-IID	1	83.51	79.05	79.05	96.61	83.81	83.81
	2	289.91	102.24	102.24	81.20	81.20	108.04
	3	275.10	121.11	92.63	102.15	102.15	102.15
	4	243.35	119.79	112.41	112.41	112.41	119.37

TABLE IV
EDGE DEVICE VERSUS ITERATION TEST ACCURACY (CIFAR-10)

Data	E \ I	Initial	1	2	3	4	5
IID	1	79.56%	82.65%	83.04%	85.02%	86.34%	87.92%
	2	75.46%	80.66%	81.10%	83.22%	85.48%	88.48%
	3	69.85%	78.19%	80.19%	82.29%	84.24%	86.40%
	4	69.77%	78.12%	81.39%	87.42%	88.42%	89.47%
Non-IID	1	62.63%	71.28%	78.28%	80.61%	81.84%	83.84%
	2	66.40%	69.48%	69.48%	70.53%	70.53%	82.86%
	3	77.13%	79.51%	81.84%	82.89%	83.59%	84.89%
	4	77.28%	79.69%	80.06%	81.06%	82.56%	85.30%

device model is trained on a distinct sample training dataset, denoted as D, comprising labels from 0 to 9. The dataset D is composed of $x\%$ of labels from set $\{0, 1, \ldots, 5\}$ and $y\%$ of labels from set $\{6, 7, 8, 9\}$, where $y > x$ ensuring the privacy of data for individual edge devices. The rationale behind such splitting and training of the initial data provides a framework for real-time data distribution, where edge devices are unaware of the type and volume of data, captured for self-training. The smaller sample size (0–5) compared with (6–9) is due to the lightweight model, having prior knowledge of the labels. This ensures that the weights do not undergo significant alterations for labels (0–5).

The learning rate and the epoch are set to 0.001 and 100, respectively, for all edge devices, along with the custom build callback function to get the optimal model for successive iterations and to meet the early stopping criteria. The initial training occurs across four edge devices, each hosting a different CNN model from the set of selected lightweight models $\in$ {Model_1,Model_2,Model_3,Model_4}, is illustrated in Figs. 4, 5, 6, and 7, respectively. The initial phase takes a bit longer time to get the best parameter, later while following the Gossip strategy, it gets easier. The training phase-2 and phase-3 are the successive training phases, indicating the impact of GL as the training accuracy increases from the previous iteration and reaches a convergence point around 20th epoch in MNIST and 25th epoch in CIFAR-10, respectively. The custom callback along with the Gossip strategy helps the model to achieve convergence with a low epoch.

D. Comparative Analysis

Tables II and III refer to the accuracy and the training time of the MNIST dataset along with Tables IV and V for CIFAR-10 with different data distributions. For the IID data, we have fixed the validation dataset to be 20% of the training dataset, while for the non-IID data, we have kept the validation dataset size unequal for different edge devices. The inverse relationship between the training time and test accuracy across different edge devices in successive iterations indicates the impact of data heterogeneity and limited straggler, making the device interact fast with the source edge device for faster convergence of all the edge devices. Fig. 5(a) illustrates the impact of Gossip in the MNIST dataset across successive iterations, shown in Fig. 5(b) and 5(c). The initial accuracy of the model deployed at the edge device having 50.63% subsequently increased to 97.82% by receiving aggregated weight from the source edge device. The similar impact can be seen across CIFAR-10, shown

TABLE V
EDGE DEVICE VERSUS ITERATION TRAINING TIME (IN SECONDS) (CIFAR-10)

Data	E \ I	Initial	1	2	3	4	5
IID	1	810.20	653.20	430.50	250.65	192.45	122.45
	2	900.05	650.25	450.30	242.38	181.25	119.44
	3	750.05	475.50	350.30	238.35	199.30	101.39
	4	650.15	360.45	275.54	190.45	120.28	99.38
Non-IID	1	583.20	309.05	279.35	116.61	105.81	100.02
	2	475.91	302.34	252.24	181.20	111.70	108.14
	3	575.10	421.11	392.63	212.15	102.15	95.15
	4	489.36	395.42	256.23	212.71	102.41	99.37

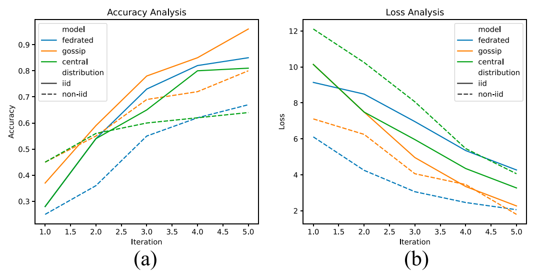

Fig. 8. Performance analysis over various data distribution. (a) Accuracy. (b) Loss.

in Fig. 6(a)–6(c) where the performance of model deployed initially having accuracy $79.56\%, 75.46\%, 69.85\%, 69.77\%$ to $87.92\%, 88.48\%, 86.42\%, 89.47\%$ till fifth iterations.

In Fig. 8, we compared the performance of the proposed GL against standard techniques such as FL and centralized learning across datasets, characterized by IID and non-IID data. Empirical results demonstrated that the proposed GL surpasses the existing ones in terms of improved accuracy by 5.02% and 6.32% compared to FL and centralized, respectively, in the case of the IID dataset. In the case of non-IID dataset, the accuracy of GL improved by 4.86% and 4.56% compared with FL and centralized, respectively. Similarly, when we observe the training loss, the GL again performed better as compared to the FL and centralized learning by reducing the loss by 5.76% and 6.97% in the case of the IID dataset and 3.54% and 2.54% in the case of the non-IID dataset. The simulation results represent the superiority of GL, particularly in scenarios featuring non-IID heterogeneous data distributions.

In Fig. 9, we compare the total requests (as a form of non-IID distribution) sent by the edge devices to the source edge device for aggregation. While comparing these requests with the standard FL and centralized learning, GL notably sends fewer requests by introducing an innovative approach by designating

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

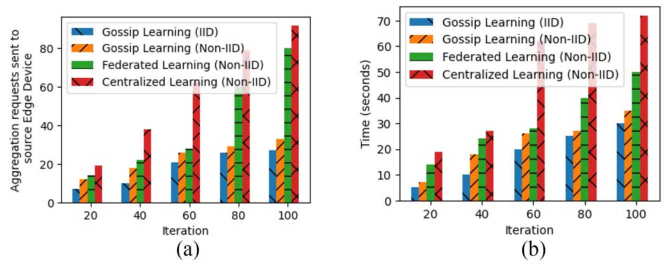

Fig. 9. Performance analysis of different models using non-IID distribution. (a) Communication overhead. (b) Latency.

a single-edge device as a source aggregator, which operates asynchronously. This unique architecture significantly reduces communication overhead by minimizing aggregation requests sent to the aggregator. As a result, latency is reduced, as there is no need to transmit data for aggregation purposes. The reduced communication overhead and latency in the proposed methodology contribute to more efficient learning processes. Furthermore, experiments show GL's effectiveness with non-IID data diversity but less distinction in performance compared with other techniques in IID scenarios. Overall, the comparative analysis highlights the substantial advantages of GL, especially in heterogeneous data environments, while showcasing the efficiency gains afforded by the novel asynchronous source aggregator architecture.

VI. Conclusion

This article introduces a new GL framework in edge networks, detailing its design, implementation, and evaluation, which substantially reduces communication overhead and handles data heterogeneity efficiently. The proposed Gossip-assisted DL strategy maximizes the utilization of data distribution, achieving a well-balanced model while improving accuracy and minimizing losses across various resource-constraint edge devices. Through GL, each edge device learns from its current model parameters and the aggregated parameters, received from the source edge device to determine the optimal model for the next iteration. Extensive simulations demonstrate GL superiority over state-of-the-art methods, such as FedAvg (FL logic) and Central algorithm, utilizing publicly available image datasets. In the future, we will enhance the model for different real-time applications for data analysis and knowledge discovery at the edge of the network with minimum delay and higher performance accuracy.

References

[1] M. Ali et al., "Edge enhanced deep learning system for large-scale video stream analytics," in *Proc. IEEE 2nd Int. Conf. Fog Edge Comput. (ICFEC)*. Piscataway, NJ, USA: IEEE Press, 2018, pp. 1–10.

[2] C.-J. Wu et al., "Machine learning at Facebook: Understanding inference at the edge," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*. Piscataway, NJ, USA: IEEE Press, 2019, pp. 331–344.

[3] J. J. Dai et al., "BigDL: A distributed deep learning framework for big data," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 50–60.

[4] C. Yu et al., "Distributed learning over unreliable networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 7202–7212.

[5] A. Kalam, S. Singh, and A. Sangal, "5G traffic forecasting using federated learning," in *Proc. 3rd Int. Conf. Secure Cyber Comput. Commun. (ICSCCC)*, 2023, pp. 629–634.

[6] L. Zhao, M. Valero, S. Pouriyeh, F. Li, L. Guo, and Z. Han, "A decentralized communication-efficient federated analytics framework for connected vehicles," *IEEE Trans. Veh. Technol.*, early access, Mar. 2024, doi: 10.1109/TVT.2024.3380582.

[7] C. Barroso-Fernández, E. Jiménez, J. L. López-Presa, M. Moreno-Cuesta, and R. Xulvi-Brunet, "Optimizing gossiping for asynchronous fault-prone IOT networks with memory and battery constraints," *IEEE Access*, vol. 12, pp. 4701–4715, 2024.

[8] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[10] Y. Wu, H. Guo, C. Chakraborty, M. R. Khosravi, S. Berretti, and S. Wan, "Edge computing driven low-light image dynamic enhancement for object detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 3086–3098, Sep./Oct. 2023.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.

[12] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2864–2880, Sep./Oct. 2023.

[13] J. Chen, J. Tang, and W. Li, "Industrial edge intelligence: Federated-meta learning framework for few-shot fault diagnosis," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3561–3573, Nov./Dec. 2023.

[14] M. Costantini, G. Neglia, and T. Spyropoulos, "FedDec: Peer-to-peer aided federated learning," 2023, *arXiv: 2306.06715*.

[15] E. T. M. Beltrán et al., "Fedstellar: A platform for decentralized federated learning," *Expert Syst. Appl.*, vol. 242, May 2024, Art. no. 122861. [Online]. Available: doi: 10.1016/j.eswa.2023.122861.

[16] G. He, C. Li, M. Song, Y. Shu, C. Lu, and Y. Luo, "A hierarchical federated learning incentive mechanism in UAV-assisted edge computing environment," *Ad Hoc Netw.*, vol. 149, 2023, Art. no. 103249. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870523001695

[17] R. Han, S. Li, X. Wang, C. H. Liu, G. Xin, and L. Y. Chen, "Accelerating gossip-based deep learning in heterogeneous edge computing platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1591–1602, Jul. 2021.

[18] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[19] H. K. Bhuyan and C. Chakraborty, "Explainable machine learning for data extraction across computational social system," *IEEE Trans. Comput. Social Syst.*, early access, May 2022, doi: 10.1109/TCSS.2022.3164993.

[20] A. Jaberzadeh, A. K. Shrestha, F. A. Khan, M. A. Shaikh, B. Dave, and J. Geng, "Blockchain-based federated learning: incentivizing data sharing and penalizing dishonest behavior," in *Proc. Int. Congr. Blockchain Appl.*, Cham, Switzerland: Springer Nature, 2023, pp. 186–195.

[21] T. H. T. Le, L. Cantos, S. R. Pandey, H. Shin, and Y. H. Kim, "Federated learning with NOMA assisted by multiple intelligent reflecting surfaces: Latency minimizing optimization and auction," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 11558–11574, Sep. 2023.

[22] B. Can, S. Soori, N. S. Aybat, M. M. Dehnavi, and M. Gürbüzbalaban, "Randomized gossiping with effective resistance weights: Performance guarantees and applications," *IEEE Trans. Control Netw. Syst.*, vol. 9, no. 2, pp. 524–536, Jun. 2022.

[23] A. Kukkar et al., "Optimizing deep learning model parameters using socially implemented IoMT systems for diabetic retinopathy classification problem," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 4, pp. 1654–1665, Aug. 2023.

[24] C.-C. Chang, C.-H. Lu, M.-Y. Chang, C.-E. Shen, Y.-C. Ho, and C.-Y. Shen, "Learning to augment graphs: Machine-learning-based social network intervention with self-supervision," *IEEE Trans. Comput. Social Syst.*, early access, Jan. 2024, doi: 10.1109/TCSS.2023.3340230.

[25] K. M. Sim, "Cooperative and parallel fog discovery and pareto optimal fog commerce bargaining," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 4, pp. 1112–1121, Aug. 2022.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on November 19,2025 at 21:51:33 UTC from IEEE Xplore. Restrictions apply.