Demo: Privacy-Preserving Decentralized Machine Learning Framework for Clustered Resource-Constrained Devices

Muhammad Ayat Hidayat
muhammad.971@s.kyushu-u.ac.jp
ISEE, Kyushu University
Fukuoka, Japan

Yugo Nakamura
y-nakamura@ait.kyushu-u.ac.jp
ISEE, Kyushu University
Fukuoka, Japan

Yutaka Arakawa
araka@ait.kyushu-u.ac.jp
ISEE, Kyushu University
Fukuoka, Japan

ABSTRACT

We present a secure decentralized learning framework suitable for resource-constrained devices within a cluster environment. Our approach focuses on enhancing privacy preservation during model aggregation by utilizing Differential Privacy. This technique adds random noise to gradients obtained from local training on edge devices before sending them for aggregation. This noise addition ensures that sensitive information within the gradients remains distorted, thus safeguarding user privacy. We showcase the implementation of our system on a cluster system employing Raspberry Pi 4 Model B devices, illustrating its feasibility and effectiveness in real-world scenarios. Through this demonstration, we highlight the practical applicability of our system in enabling secure decentralized learning within resource-constrained environments.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

KEYWORDS

Privacy, Decentralized Learning, Differential Privacy, Resource-Constrained

ACM Reference Format:
Muhammad Ayat Hidayat, Yugo Nakamura, and Yutaka Arakawa. 2024. Demo: Privacy-Preserving Decentralized Machine Learning Framework for Clustered Resource-Constrained Devices. In *The 22nd Annual International Conference on Mobile Systems, Applications and Services (MOBISYS '24), June 3–7, 2024, Minato-ku, Tokyo, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3643832.3661843

1 INTRODUCTION

Decentralized machine learning (DML) represents a form of machine learning wherein data is dispersed across numerous nodes within a network rather than being centralized in one location. This distribution enhances scalability and adaptability. Moreover, DML enhances security and privacy; instead of transmitting raw data to another edge device or central server, only local parameters derived from local training are sent for aggregation. This ensures that raw data remains confined to local device storage, thus safeguarding it against adversarial attacks. However, despite the absence of raw data sharing within the network, local parameters are still susceptible to sniffing attacks, potentially resulting in the theft of model parameters. This could facilitate an inference attack to extract sensitive information from the local training data stored on edge devices [1]. Several solutions have been proposed to provide additional protection for DML, such as Blockchain [3], Homomorphic Encryption (HE) [5], or Differential Privacy (DP) [2]. However, implementing HE requires extra computing resources, blockchain requires large storage, and DP increases the risk of communication bottleneck, rendering these solutions unsuitable for edge devices with limited computing and storage resources [4].

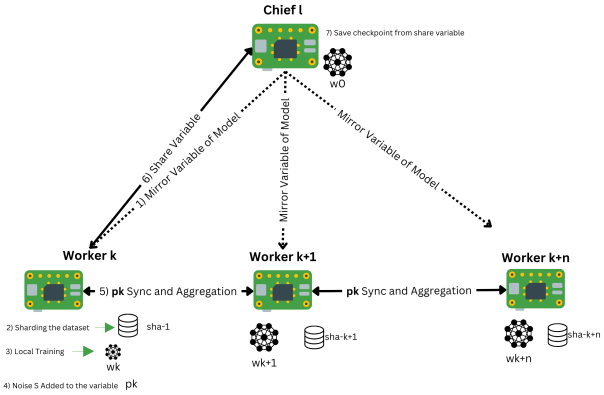

Figure 1: System overview of the system

This demo presents a system that establishes a secure process for DML. By leveraging the benefits of DP, our approach offers a promising solution for securing DML on-edge devices with limited resources. Furthermore, the system is implemented in cluster scenarios, where participating edge devices are grouped, and the dataset is sharded for training, ensuring scalability, particularly with large datasets. Specifically, our system is capable of :

(1) Protecting local parameters that will be exchanged between edge devices for aggregation by distorting the original value with noise
(2) Reducing communication bottlenecks that occur in the communication process
(3) Making the training process efficient
(4) Being implemented in devices with limited resources

2 SYSTEM OVERVIEW

We consider this system a DML system operating within a cluster scenario comprising two nodes: a chief node and a worker node, as illustrated in Fig. 1. The chief node does not control the learning

MOBISYS '24, June 3–7, 2024, Minato-ku, Tokyo, Japan

ACM ISBN 979-8-4007-0581-6/24/06
https://doi.org/10.1145/3643832.3661843

process or aggregate local parameters from the worker node; instead, it solely stores checkpoints that a new worker node can later access to prevent it from recommencing the training process from the beginning. The main process will be executed by the worker node. This process included dataset sharding, local training, noise addition, aggregation, and synchronization. Here is the detailed process of our proposed system :

(1) **Intialization**. In this step, the chief node shares the latest checkpoint $w0$ with all workers participating in the learning process. this included model variables and parameters.
(2) **Dataset sharding**. In this step, each worker is assigned a subset *sha* of the entire dataset, ensuring that each worker k operates on a distinct data set during processing. This process minimizes redundant computations, reducing the computational resources required for the training process.
(3) **Local Training**. In this step, the worker starts local training using the dataset assigned by the previous process. This process generates local model parameters wk.
(4) **Noise Addition**. In this step, local parameters generated from local training will have noise added to them. The amount of noise added is proportional to the standard deviation of the clipping value, which is adjusted for every iteration and then multiplied by the noise multiplier. This process generates noisy local parameters pk that will be shared with another worker in the cluster.
(5) **Aggregation**. In this step, each worker aggregates noisy local parameters from other workers, and the aggregated results are used to update the local model.
(6) **Synchronization**. In this step, the worker shares its updated local model with another worker in the cluster for synchronization. This process ensures that all workers are aligned in their progress and state, maintaining consistency. Additionally, the updated local model will also be shared with the chief node and saved as checkpoints.

3 SYSTEM IMPLEMENTATION

We implemented our system in real-world scenarios using physical devices. To represent resource-constrained devices, we utilized Raspberry Pi 4 Model B equipped with a quad-core Cortex-A72 processor, 4 GB of memory, and 50 GB of storage. The implementation involved 16 Raspberry Pi devices organized into four clusters, each consisting of four Raspberry Pi devices. Within each cluster, one Raspberry Pi device acted as the chief node, responsible for saving and sharing the latest checkpoint between workers in the cluster, while the remaining three Raspberry Pi devices served as worker nodes, performing the actual training tasks. To facilitate communication and coordination between clusters, all clusters were interconnected using a local area network (LAN). The detailed infrastructure of our system can be seen in Fig. 2.

4 DEMONSTRATION

We demonstrate our system to simulate mental health prediction. We run this simulation using the mental health open dataset from Kaggle[1]. We will use 50 epochs and 10 steps for the training parameter, a learning rate of 0.001, and a clip value of 0.1. For the model, we will use CNN with the Softmax activation method. For the demonstration, we will divide the process into 2 configurations:

(1) **Without malicious worker**. In this configuration, we will start the learning process without malicious workers. Every worker honestly does local training without a modified dataset.
(2) **One malicious worker per cluster**. In this configuration, we assume that each cluster includes one malicious worker with a modified dataset. The dataset on this malicious worker has already undergone label flipping, rendering the classes poisonous.

We will display the training results on the website, enabling attendees to compare accuracy and resource usage across different configurations. This will enable them to observe the performance of our proposed system under adversarial attacks on the model. Attendees can also interact with the system by selecting the cluster containing the malicious worker and choosing training parameters and batch size for local training.

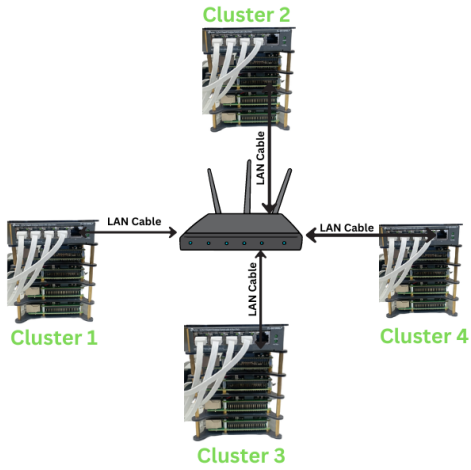

Figure 2: Infrastructure of the system

[1]https://www.kaggle.com/code/jagannathrk/mental-health-prediction/input

REFERENCES

[1] Ehsan Hallaji, Roozbeh Razavi-Far, Mehrdad Saif, Boyu Wang, and Qiang Yang. 2024. Decentralized Federated Learning: A Survey on Security and Privacy. *IEEE Transactions on Big Data* 10, 2 (2024), 194–213. https://doi.org/10.1109/TBDATA.2024.3362191
[2] Muhammad Ayat Hidayat, Yugo Nakamura, and Yutaka Arakawa. 2024. Privacy-Preserving Federated Learning With Resource-Adaptive Compression for Edge Devices. *IEEE Internet of Things Journal* 11, 8 (2024), 13180–13198. https://doi.org/10.1109/JIOT.2023.3347552
[3] Caner Korkmaz, Halil Eralp Kocas, Ahmet Uysal, Ahmed Masry, Oznur Ozkasap, and Baris Akgun. 2020. Chain FL: Decentralized Federated Machine Learning via Blockchain. In *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*. 140–146. https://doi.org/10.1109/BCCA50787.2020.9274451
[4] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640. https://doi.org/10.1016/j.future.2020.10.007
[5] Mengxue Shang, Dandan Zhang, and Fengyin Li. 2023. Decentralized Distributed Federated Learning Based on Multi-Key Homomorphic Encryption. In *2023 International Conference on Data Security and Privacy Protection (DSPP)*. 260–265. https://doi.org/10.1109/DSPP58763.2023.10405290