Decentralized Machine Learning with qLORA, Storj, and Cardano Hydra: A Secure and Efficient Framework

Abstract

This paper introduces a decentralized machine learning framework that integrates Quantized Low-Rank Approximation (qLORA) for efficient model training, Storj for secure decentralized storage using satellites and buckets, and Cardano Hydra for state channel communication. The system addresses key challenges in distributed learning such as computational efficiency, data privacy, and secure communication. The entire workflow, including secure multi-party computation and cryptographic protocols like Paillier encryption, is formally verified using TLA+ to ensure correctness. We demonstrate the feasibility of our approach with experiments on the MNIST dataset, showing the scalability and robustness of the solution.

1. Introduction

With the increasing size and complexity of machine learning models, decentralized training methods are essential to preserve data privacy and minimize communication overhead. This paper proposes a solution combining qLORA for computational efficiency, Storj for decentralized storage, and Cardano Hydra for secure communication. By incorporating advanced cryptographic techniques like Zero-Knowledge Proofs (ZKP) and Paillier encryption, our system ensures data privacy during model training. Furthermore, the system is formally verified using TLA+, ensuring correctness and security in all operations.

Mathematical Problem Formulation

In decentralized learning, the goal is to collaboratively train a global machine learning model MM across multiple participants, each with their own local dataset Xi The global loss function to be minimized is:

$$\min_{w} L(w) = \frac{1}{n} \sum_{i=1}^{n} L_i(w)$$

where w represents the global model weights, and $L_i(w)$ is the local loss function based on the dataset X_i.

2. Background

2.1 Quantized Low-Rank Approximation (qLORA)

qLORA reduces computational overhead by using low-rank approximations and quantization to compress the weight matrices of large models. This technique is particularly beneficial in decentralized environments where participants may have limited computational resources.

Mathematical Model for qLORA

> Given a weight matrix $W \in \mathbb{R}^{m \times n}$, qLORA approximates W using two smaller matrices $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$, where $k \ll \min(m, n)$:
>
> $$W \approx A \times B$$

The entries of A and B are then quantized to reduce storage and computation requirements. The challenge is to find an optimal trade-off between compression and accuracy.

2.2 Distributed Learning

In decentralized learning, multiple participants collaboratively train a global model while keeping their data local. At each iteration, participants update the global model by sharing their local gradients or model updates. A common aggregation method is **Federated Averaging (FedAvg)**, where the global model weights are updated as:

$$w_{\text{global}} = \frac{1}{n} \sum_{i=1}^{n} w_i$$

This approach ensures that no raw data is exchanged, preserving privacy.

2.3 Storj: Decentralized Storage Using Satellites and Buckets

Storj is a decentralized cloud storage solution that leverages a network of independent storage nodes coordinated by **satellites**, which handle metadata and access management. Data is stored in **buckets**, which are logical containers for storing objects such as model weights and gradients.

Mathematical Model for Storj Satellites and Buckets

Let S_j represent the satellite coordinating the storage for participant j, and B_j represent the bucket associated with that participant, where the encrypted model updates or gradients are stored. The data is encrypted before being uploaded:

$$E(G_j) = \text{Encrypt}(G_j, pk)$$

where G_j is the gradient computed by participant j, and pk is the shared public key for encryption. The satellite S_j ensures that only authorized participants can access the data stored in B_j, ensuring privacy and security.

2.4 Cardano Hydra: Scalable State Channels

Cardano Hydra is a state channel solution that enables secure and efficient communication between participants during decentralized training. Hydra allows off-chain interactions that are later settled on the Cardano blockchain, reducing the cost and latency of on-chain transactions.

3. Methodology

Our decentralized machine learning framework integrates qLORA for model compression, Storj for decentralized storage, and Cardano Hydra for secure communication. We also employ advanced cryptographic techniques such as Paillier encryption and Zero-Knowledge Proofs (ZKP) to ensure privacy and security during the collaborative training process.

3.1 qLORA for Efficient Training

qLORA compresses model weights through low-rank approximation and quantization. This allows each participant to train locally with reduced computational overhead. The compressed weights are updated and shared among participants using secure communication channels.

3.2 Secure Multi-Party Computation Using Zero-Knowledge Proofs (ZKP)

ZKPs allow participants to prove the correctness of their encrypted gradients without revealing the actual data. Each participant j computes their gradient Gj locally, encrypts it using Paillier encryption, and then generates a ZKP to prove the correctness of the encryption:

$$E(G_j) = \mathbf{PaillierEncrypt}(G_j, pk)$$

The ZKP ensures that Gj is valid without revealing Gj.

The ZKP setup uses sigma protocol.

1. **Initial Setup**:
   - Let G be a cyclic group of prime order p, with generator g.
   - The prover P has a secret x, and wants to prove knowledge of x without revealing it.
   - The verifier V knows the public value $h = g^x$.

2. **Protocol Steps**:
   1. **Commitment**:
      - The prover P selects a random $r \in \mathbb{Z}_p$, and computes the commitment $t = g^r$.
      - P sends t to V.

      $$t = g^r$$

   2. **Challenge**:
      - The verifier V sends a random challenge $c \in \mathbb{Z}_p$ to P.

      $$c \in \mathbb{Z}_p$$

   3. **Response**:
      - The prover P computes the response $s = r + c \cdot x \mod p$, and sends s to V.

      $$s = r + c \cdot x \mod p$$

   4. **Verification**:
      - The verifier checks that the following equality holds:

      $$g^s \stackrel{?}{=} t \cdot h^c$$

If the equality holds, the verifier is convinced that the prover knows x.

3.3 Paillier Encryption for Secure Gradient Aggregation

Paillier encryption is a homomorphic encryption scheme that enables secure aggregation of encrypted gradients. Given two encrypted gradients $E(G_i)$ and $E(G_j)$, Paillier allows the summation of encrypted values:

$$E(G_i) + E(G_j) = E(G_i + G_j)$$

This property allows gradients to be aggregated securely without needing to decrypt them until after aggregation.

Key Generation and Encryption

Participants generate a public-private key pair (pk, sk). The encryption of a gradient G_j under Paillier encryption is defined as:

$$E(G_j) = (1 + nG_j)r^n \mod n^2$$

where $r \in \mathbb{Z}_n^*$ is a random value, and n is a public key parameter.

3.4 Secret Sharing: Additive and Shamir

Additive Secret Sharing

In additive secret sharing, a secret s is split into n shares $s_1, s_2, \ldots, s_n$, where the secret can be reconstructed by summing the shares:

$$s = \sum_{i=1}^{n} s_i \mod p$$

This method is computationally efficient and works well for large numbers.

Shamir's Secret Sharing

Shamir's secret sharing encodes the secret s as a constant in a polynomial $P(x)$. Each participant receives a point $(x_i, P(x_i))$, and the secret is reconstructed using Lagrange interpolation if enough shares are provided:

$$s = \sum_{i=1}^{t} P(x_i) \prod_{j \neq i} \frac{x_j}{x_j - x_i}$$

3.5 Workflow

1. **Initialization**: Participants agree on training parameters and initialize a Hydra state channel for secure communication.
2. **Local Training**: Each participant jj trains the model on their local data, computes gradients Gj, and encrypts them using Paillier encryption.
3. **Storage and Sharing**: Encrypted gradients E(Gj)E(Gj) are stored in each participant's Storj bucket BjBj, coordinated by the Storj satellite SjSj.
4. **Secure Aggregation**: Gradients are aggregated homomorphically using Paillier encryption and then decrypted using secret sharing.
5. **Formal Verification**: The entire process, including communication and aggregation, is formally verified using TLA+.

4. Experimental Setup

We implemented the system using the MNIST dataset, where each participant trains a model locally and shares encrypted updates through Hydra and Storj. An experiment with finetuning using qLora was also conducted.

4.1 Hardware and Software

- For MNIST
    - One x86 PC 16Gb RAM for running multiple parties
    - 3 Raspberry Pi for running Hydra
- qLora
    - 3 desktops with 16GB RAM and 8-core CPUs with Nvidia GPU
    - 3 Raspberry Pi for running Hydra
- PyTorch for model training.
- Storj for decentralized storage.
- Hydra for secure communication.

5. Results

The system maintained high accuracy as the number of participants increased, demonstrating the scalability and efficiency of the decentralized learning framework:

$$w_{\text{global}} = \frac{1}{n} \sum_{i=1}^{n} w_i$$

Number of Participants	Accuracy	Baseline Accuracy
2	94.8%	91%
4	95.0%	92%
10	99.0%	91.8%

The formal verification with TLA+ ensured that all aspects of the system, including storage, communication, and aggregation, are correct and resilient

6. Conclusion and Future Work

This paper presents a decentralized learning framework that integrates qLORA, Storj, and Cardano Hydra for secure and efficient machine learning. The use of cryptographic techniques like Paillier encryption and ZKPs ensures data privacy, while the system's resilience and correctness are confirmed through TLA+ verification.

Future work will explore integrating NFTs and escrow contracts to enforce stricter trust mechanisms between participants. Additionally, more advanced machine learning models will be tested in this decentralized setup.

References

1. Cardano Hydra: Hydra.family
2. Storj documentation: https://docs.storj.io
3. QLoRA: Efficient Finetuning of Quantized LLMs: Tim Dettmers, et al.