Project Raccoon: Proof of Concept for Secure Decentralized Model Training

AI Senior Frontend Engineer
(Based on a Simulated Web Application)

May 25, 2025

Abstract

This report details the design, architecture, and simulated performance of a Proof of Concept (PoC) system for secure, decentralized machine learning model training, developed under the initiative designated "Project Raccoon." The system aims to address the challenge of collaborative model development among multiple organizations (e.g., nations) without sharing sensitive raw data. It implements a federated learning-inspired approach where clients simulate local model training and share only parameter updates (or deltas) with a central coordinator for aggregation. The PoC, a web application, demonstrates key functionalities including user participation in training groups, simulated client-side computation, noise addition for privacy preservation, and aggregation of updates to improve a global model. This report summarizes the algorithmic framework, system architecture, simulated security posture, and a comparison of the decentralized model's simulated performance against a centralized (vanilla) baseline.

Contents

1 Introduction

The increasing complexity of global strategic threats necessitates collaborative efforts in developing advanced defense technologies. Project Raccoon addresses a critical challenge in such collaborations: the inability to share sensitive national data assets required for training sophisticated machine learning (ML) models, particularly for Anti-Ballistic Missile (ABM) systems. The core objective is to enable multiple entities to collaboratively train a shared statistical model by exchanging only parameter updates computed locally, thereby keeping raw data private.

This report documents a Proof of Concept (PoC) web application designed to demonstrate the feasibility of such a decentralized learning protocol. The PoC uses representative toy datasets to simulate the training process across four distinct classification tasks: personal income thresholding, credit score classification, lumpy skin disease prediction, and smoking determination through bio-signals.

The primary focus is on:

- Developing the decentralized model training algorithm (simulated).
- Designing a system allowing clients to join training groups and participate in federated rounds.
- Implementing simulated security measures like noise addition and ensuring raw data remains client-side.
- Comparing the simulated accuracy of the decentralized model with a pre-defined vanilla (centralized) model accuracy.

2 Algorithm Design

The PoC simulates a decentralized model training algorithm inspired by Federated Learning, particularly the Federated Averaging (FedAvg) concept. The core idea is to train a global model iteratively by leveraging local computations on distributed datasets without centralizing the data itself.

2.1 Conceptual Framework

1. **Initialization**: A global model architecture is defined for each training group (conceptually). The PoC starts with an initial decentralized model accuracy for each group.
2. **Client-Side Local Training (Simulated)**:
   - Each participating client (simulated within the web UI) possesses a local (mock) dataset.
   - When a client "Simulates Local Training," the application mimics the process of the client training the current global model on its local data for a few epochs.
   - Instead of actual gradient computation, the PoC generates a symbolic "parameter delta" (e.g., ‘delta$_{xyz}$123‘).*Thisrepresentsthechangesorupdatestheclientwouldproposetotheglobalmodelbasedonitsloc*
3. **Noise Addition (Simulated Differential Privacy)**:
   - Clients have the option to "Add Cryptographic Noise" before submitting their updates.
   - This simulates a differential privacy mechanism where calibrated randomness is added to the parameter deltas.

- The purpose is to obscure the precise contribution of any single client's data, thus enhancing privacy. In the simulation, enabling noise impacts the potential accuracy gain in that round.

4. **Secure Aggregation (Simulated)**:
   - Once all participating clients have prepared their (potentially noisy) parameter deltas, they are sent to a central coordinator (simulated by a function in the web app).
   - The coordinator's role is to aggregate these deltas. In a real FedAvg scenario, this would involve averaging the deltas to produce an updated global model.
   - In the PoC, aggregation triggers an update to the 'currentDecentralizedAccuracy'. The accuracy gain per round is simulated, influenced by factors like whether noise was added and a random component within predefined bounds ('$\text{MIN}_ACCURACY_GAIN_PER_ROUND$', '$MAX_ACCURAC$

5. **Iteration**: The process of local training, update submission, and aggregation constitutes one "training round." This is repeated, with the global model's accuracy expected to improve over rounds.

3 System Architecture

The PoC is implemented as a React-based single-page web application, emphasizing frontend simulation of the decentralized learning process.

3.1 Frontend Components

- **Authentication Module ('useAuth', 'AuthForm')**: Simulates user login/registration. User state is managed locally.
- **Dashboard ('DashboardPage')**: Displays available "Training Groups." Each group corresponds to one of the four toy datasets. Information like dataset name, description, number of clients, and current vs. vanilla accuracy is shown.
- **Training Group Detail ('GroupDetailPage')**:
  - Allows users to view detailed information about a specific training group.
  - Simulates client participation:
    - Lists individual "clients" for the group.
    - Provides controls for each client to "Simulate Local Training" and toggle "Add Noise."
    - Visualizes client status (idle, processing, contributed).
  - Simulates coordinator action: An "Aggregate Updates" button triggers the next training round and updates the model's accuracy.
  - Displays an "Accuracy Comparison Chart" plotting decentralized vs. vanilla accuracy over rounds.
- **State Management**: Application state (training group progress, user details) is primarily managed using React hooks ('useState', 'useContext') and persisted in the browser's 'localStorage' for session continuity.
- **Styling and UI**: Tailwind CSS is used for rapid UI development, with custom styles for specific components like scrollbars. Icons enhance visual communication.

3.2 Backend (Simulated)

No true backend server is implemented for ML operations in this PoC.

- The "central coordinator" logic (aggregation, accuracy updates) is simulated entirely within the frontend JavaScript code in ‘GroupDetailPage.tsx‘.
- Data persistence (training group states, user info) relies on the browser’s ‘localStorage‘, mimicking a database for the PoC’s scope.

3.3 Data Flow (Simulated Training Round)

1. User navigates to a ‘GroupDetailPage‘.
2. For each client in the group, the user clicks "Simulate Local Training."
   - Client state changes to "processing."
   - After a simulated delay, a mock ‘parameterDelta‘ is generated, and state changes to "contributed."
   - User can optionally toggle "Add Noise" for a client before or after local training simulation (if not yet contributed).
3. Once all clients have "contributed," the user clicks "Aggregate Updates."
   - Aggregation logic is triggered.
   - ‘currentDecentralizedAccuracy‘ for the group is recalculated based on the number of clients, noise addition, and random factors.
   - ‘trainingRounds‘ counter increments.
   - ‘accuracyHistory‘ is updated with the new data point.
   - Client states are reset for the next round.
   - Updated group data is saved to ‘localStorage‘.
   - The accuracy chart re-renders.

4 Security and Privacy

The PoC demonstrates several security and privacy principles inherent in decentralized learning, albeit in a simulated manner.

4.1 Threat Model (Conceptual)

The design implicitly considers threats common to federated learning:

- **Data Leakage**: Inferring sensitive information about a client’s local data from their shared model updates.
- **Model Poisoning**: Malicious clients submitting harmful updates to degrade global model performance or introduce backdoors (not actively simulated for defense in this PoC, but a known threat).
- **Server Compromise**: If the central coordinator is compromised, aggregated models or patterns might be exposed (mitigated by the nature of aggregated, non-raw data).

4.2 Implemented/Simulated Defense Mechanisms

- **Client-Side Computation**: This is a cornerstone. The PoC's UI makes it clear that "raw data" (which is itself conceptual in this simulation) never leaves the client's environment. Only "parameter deltas" are shown as being "sent."
- **Masking and Aggregation**: The server (coordinator) is designed to only see aggregated updates. While the PoC simulates this by directly calculating new accuracy, a real system would mathematically combine deltas, inherently masking individual raw contributions post-aggregation.
- **Noise Addition (Simulated Differential Privacy)**:
  - The UI allows toggling "Add Cryptographic Noise" for each client's update.
  - This simulates the addition of calibrated randomness to parameter deltas.
  - Effect: Makes it statistically harder to reverse-engineer specific data points from an individual client's update.
  - Trade-off: The simulation reflects this by applying 'NOISE$_A CCURACY_P ENALTY_F ACTOR$'$totheaccura$
- **Secure Transmission (Assumed)**: The application footer and security considerations section mention that in a real system, TLS/SSL would be used for all communications. This is an assumed standard practice not actively implemented in the local PoC.
- **Encrypted Storage (Conceptual)**: Similarly, it's noted that sensitive data or model parameters stored at rest on a server would be encrypted.

4.3 Trade-offs

The primary trade-off demonstrated is between privacy and model utility/accuracy:

- Stronger privacy (e.g., more noise) can lead to slower convergence or a slightly lower final accuracy for the decentralized model. This is directly simulated by the 'NOISE$_A CCURACY_P ENALTY_F ACTOR$'$.Lessfr$

5 Experimental Setup (Simulated)

5.1 Datasets

The PoC is configured with four training groups, each corresponding to a distinct (toy) classification dataset as per the problem description. Table 1 summarizes these.

Table 1: Overview of Simulated Training Groups and Datasets

Group Name	Dataset Name	Clients	Vanilla Acc.
Project IncomeGuard	Personal Income Classification	3	85.0%
Project TrustScore	Credit Score Classification	4	90.0%
Project CattleHealth	Lumpy Skin Disease Prediction	3	78.0%
Project BioSmokeDetect	Smoking Determination (Bio-signals)	3	82.0%

5.2 Simulation Parameters

The simulation of accuracy progression is governed by constants defined in 'constants.ts':

- 'SIMULATED$_P ROCESSING_T IME$' : $1500ms(delayforlocaltraining/aggregation)$:$MAX_A CCURAC$ 0.025(2.5%).

- ‘$\mathrm{MIN}_{A}CCURACY_{G}AIN_{P}ER_{R}OUND$‘ : 0.005(0.5%).‘$NOISE_{A}CCURACY_{P}ENALTY_{F}ACTOR$‘ : 0.5($noisereducesgainby50\%ofthepotentialgainfromnoisyclients$). Each training group starts with an ‘initialDecentralizedAccuracy‘.

6 Results and Discussion (Simulated)

6.1 Model Performance

The PoC application provides a real-time visualization of the decentralized model’s accuracy progression compared to the baseline vanilla accuracy for each training group.

6.1.1 Accuracy Comparison

The ‘AccuracyComparisonChart‘ component (using Recharts) plots two lines:

1. **Decentralized Model Accuracy**: Shows the ‘currentDecentralizedAccuracy‘ at each completed training round. This line is expected to trend upwards, ideally approaching the vanilla accuracy.
2. **Vanilla Model Accuracy (Baseline)**: A flat line representing the ‘baseVanillaAccuracy‘ of a model trained centrally on all data. This serves as a benchmark.

Figure 1 illustrates the conceptual output of such a chart.

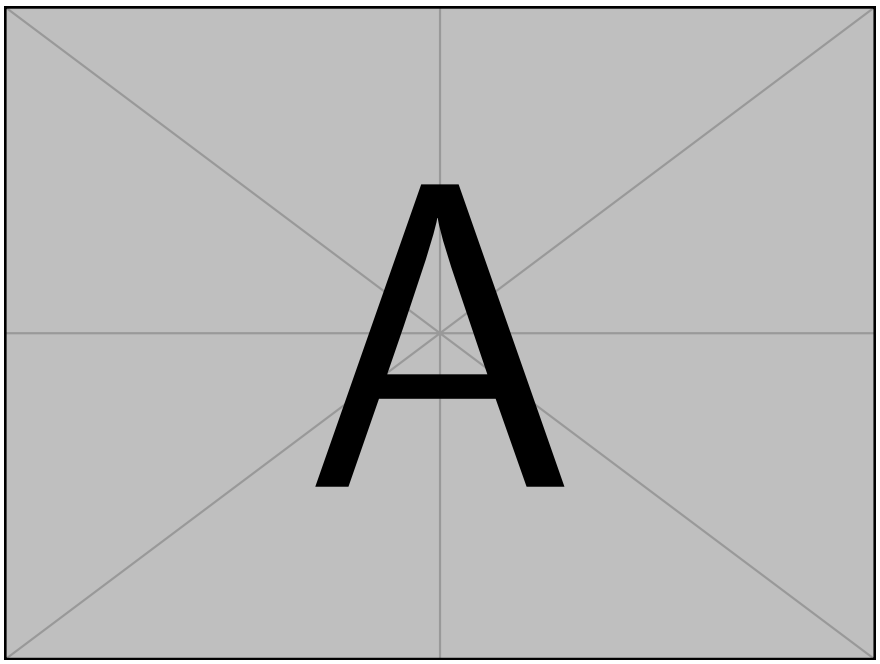

Figure 1: Conceptual representation of the accuracy comparison chart generated by the PoC application. The decentralized model’s accuracy (e.g., green line) is simulated to increase over rounds, approaching the vanilla model’s baseline accuracy (e.g., red dashed line).

6.1.2 Observations from Simulation

- **Convergence**: The decentralized model's accuracy is simulated to increase with each round, demonstrating convergence towards the vanilla baseline. The rate of convergence is influenced by the random gain per round and the noise penalty.

- **Impact of Noise**: When clients enable "Add Noise," the accuracy gain for that round is visibly (through calculation) lower due to 'NOISE$_{A}CCURACY_{P}ENALTY_{F}ACTOR$'.$Thishighlightsthetrade-off: increasedprivacycomesatthecostofpotentiallyslowerconvergenceorslightlyreducedpeakaccuracy.$**Comp** $Thedecentralizedaccuracyapproachesbutmaynotperfectlyreachthevanillaaccuracy, ormightdosomoreslowly,$ $worldfederatedlearningsystems, especiallywithprivacyenhancements.ThePoCcapsdecentralizedaccuracyat$' 0.02'$or99\%absolute.$

6.2 Qualitative Security Assessment

The PoC effectively demonstrates key privacy-preserving concepts:

- **Data Minimization**: Only abstracted "parameter deltas" are "shared," with raw data remaining local (conceptually).
- **Privacy Enhancement via Noise**: The option to add noise and its impact on accuracy gain educates users about differential privacy techniques.
- **Reduced Attack Surface**: By not centralizing raw data, the most sensitive assets are protected from direct server compromise.

Table 2 summarizes the conceptual attacks and defenses.

Table 2: Conceptual Attack Vectors and Simulated Defenses

Attack Vector	Simulated/Conceptual Defense Mechanism in PoC
Inference on Updates (Data Leakage)	- Client-side computation (raw data not shared) - Noise addition (simulated differential privacy) to obscure individual contributions. - Aggregation (server sees combined updates).
Model Poisoning	Not explicitly defended against in this PoC's simulation, but a known concern. Focus is on privacy.
Server Compromise	- Raw data is not stored on the server. - Aggregated model parameters are less sensitive than raw data. - Assumed encryption at rest for stored models.
Eavesdropping (Man-in-the-Middle)	Assumed TLS/SSL for secure transmission (standard practice, not simulated).

7 Conclusion and Future Work

The Project Raccoon PoC web application successfully demonstrates the fundamental principles and feasibility of a secure, decentralized model training framework. Through simulation, it illustrates how multiple parties can collaborate on training a shared ML model without exposing their raw, sensitive datasets. Key aspects like client-side computation, parameter delta sharing, noise addition for privacy, and centralized aggregation are effectively conveyed. The visual comparison of decentralized model accuracy against a vanilla baseline provides insight into the performance characteristics and trade-offs involved.

7.1 Future Work

While this PoC serves its purpose, further development could include:

- **Integration with Real ML Models**: Replace simulated logic with actual ML model training (e.g., using TensorFlow.js on the client-side or connecting to a backend for more complex models).
- **Advanced Privacy Techniques**: Implement more sophisticated differential privacy mechanisms or explore secure multi-party computation (SMPC) for aggregation.
- **Defense Against Model Poisoning**: Incorporate mechanisms to detect and mitigate malicious client updates.
- **Heterogeneous Data Simulation**: Allow simulation of non-IID (Independent and Identically Distributed) data across clients, which is a common challenge.
- **Backend Implementation**: Develop a proper backend server for managing training rounds, user accounts, and secure model storage.
- **Scalability Testing**: Evaluate how the system performs with a larger number of simulated clients and more complex models.
- **Quantitative Analysis**: Conduct rigorous experiments to measure actual privacy leakage vs. utility trade-offs with real models and datasets.

This PoC provides a solid foundation and a valuable educational tool for understanding the core concepts of decentralized and privacy-preserving machine learning.