Federated Learning Protocol Secured by EigenLayer

Introduction

This post outlines the general idea behind a Federated Learning Protocol secured by EigenLayer, focusing on data privacy, data compensation, and the use of distributed computing resources in the context of large language model (LLM) training and inference. The proposed protocol aims to enable users to collaboratively train an on-chain Foundational Large Language Model (yes, like ChatGPT…) using computational resources available off-chain (Local Model training) and private data available off-chain (locally available data).

Protocol Architecture

The Federated Learning Protocol would be implemented in the following stages:

1. Global Model Initialization:

A Foundational Large Language Model (hereafter referred to as the “Global Model”) is initialized and its starting parameters (weights and biases) are stored through EigenDA or another similar data availability layer. The Global Model serves as a starting point for Local Model training.

1. Global Model Parameter Sharing:

The parameters of the Global Model are shared with all restakers who have opted into the Federated Learning protocol.

1. Local Model Training by Restakers:

Each restaker obtains the parameters from the Global Model and uses them to initialize their Local Model. Opted-In restakers then train this Local Model on their own local data. Note that, while each restaker’s Local Model has the same architecture and size as the main Global Model that lives on Ethereum (or an L2), the key difference lies in the size of the dataset used for training; since each restaker locally trains the latest version of the Global Model on only their local data (rather than on the entire data set used to create the Global Model), the training process for Local Model training is vastly less computationally intensive. This is a fundamental principle of federated learning and makes it more feasible for restakers to train on their local hardware. Furthermore, it is worth noting that, with the continued exponential advancements in open-source LLM compression in recent months, it is anticipated that the computational and capital resources required to train a powerful LLM locally, will continue to go lower. For example, see Alpaca 7B, which was released by Stanford just last week, is as powerful as DaVinci-3 and costs less than $600 to train, with no special hardware and no high-end graphics card.

1. Local Model Parameter Updates:

After local training on a given restaker’s machine is complete, each restaker computes an update to the Local Model parameters based on the local training results. This update is represented as a gradient. For those not familiar with deep learning and the gradient descent optimization algorithm, the gradient is a vector of partial derivatives with respect to the model parameters (i.e., the weights and biases). These gradients represent the changes needed in the model parameters to minimize the model’s loss function, which quantifies the difference between the model’s predictions and the actual target values in the training data. In the context of this proposed Federated Learning Protocol, the restaker encrypts their newly computed gradient vector using an additive secret sharing scheme that involves splitting their locally computed gradient into multiple random shares, which, when combined, reveal the original gradient. Additionally, the restaker submits a zero-knowledge proof (ZKP) with their gradient vector result to prove that he/she/it has followed the training procedure prescribed in the protocol’s slashing contracts and has computed the model gradients accurately, while keeping the data used in the training process private.

1. Aggregation:

The restaker sends one share of their split gradient to each of the other restakers participating in the aggregation process (the protocol can be designed such that all or only some of the restakers participate in parameter aggregation). Each restaker participating in aggregation should now have one share of every other restaker’s gradient. Every restaker then sums the shares they have received from other restakers. This results in a new share, which is a partial sum of the encrypted gradients. These partial sums are then sent to the Aggregator (a smart contract or a designated restaker). The Aggregator combines the partial sums to obtain the aggregated gradient, while ensuring that individual updates received from each restaker is not exposed during the aggregation process.

1. Global Model Update:

The Aggregator uses the aggregated gradient to update the Global Model parameters on-chain, resulting in an improved version of the Global Model.

The sharing of the updated Global Model parameters (Step 2

above) is then repeated and the process for updating Global Model parameters of the Federated Learning Large Language Model is repeated in perpetuity, ever improving from the constant addition of new private data on which restakers locally train.

Protocol Participants

The proposed Federated Learning Protocol will consist of the following participants:

EigenLayer Restakers:

ETH validator nodes responsible for providing computational resources for training the Global Model locally off-chain using a vastly smaller dataset than that used to produce the Global Model's parameters. Restakers are compensated for making use of their computational resources and updating the state of the Global Model's on-chain parameters.

Aggregator:

A designated validator node or smart contract responsible for securely aggregating Local Model gradient updates from restakers to make a state update to the on-chain Global Model parameters.

Global Model Users:

Individuals and entities that interact with the on-chain Global Model in exchange for a usage fee.

Compensation for Computational Resources and Slashing Risk

As with all middleware services built on EigenLayer, Restakers would receive ETH compensation for the computational resources they provide to the protocol, such as processing power, memory, and storage, needed to continually improve the Global Model's loss function over time, and to compensate for the risk of unintended or malicious slashing.

Compensation for Private and Proprietary Information

Aside from being compensated for the critical function of providing computational resources, arguably more interesting would be the potential to compensate Restakers for the value of their locally stored private data/information that they have access to for Local Model training.

Individuals and entities have access to an inordinate amount of personal and/or proprietary information, that is highly private or gives them some sort of competitive advantage over their peers. This information is often stored locally on individuals' and entities' devices and/or is shared only with specific trusted third parties.

To incentivize Restakers to include locally stored high-quality, private, and/or proprietary data/information in their Local Model training dataset, the Federated Learning Protocol could implement a fingerprinting system whereby Restakers can be further compensated based on the influence their gradient has on the Global Model update and can do so in a way that still preserves the privacy of their data/information.

To implement this Data Compensation System, the protocol would need to:

1. Generate a unique "fingerprint" for each Restaker when they opt into the protocol. This fingerprint could be derived from the Restaker's wallet address.

2. Attach the fingerprint to that restaker's gradient submission when the restaker sends the updated gradient calculated by their Local Model to the Aggregator, including the fingerprint as metadata along with the gradient update. Note that the gradient computed by the restaker is based on whatever private data they elect to train on, and so their computed gradient inherently serves as a proxy for the data they elect to include in training their Local Model version.

3. The Aggregator receives gradient updates from restakers and associates each gradient update with the corresponding fingerprint, aggregating the updates in such a way that individual gradients are not exposed during the process. During the aggregation process, the fingerprints associated with each gradient update are preserved, while ensuring that the privacy of the gradients is maintained during secure aggregation.

4. Once the aggregation process is completed, the Aggregator updates the on-chain parameters of the Global Model with the aggregated parameters from the culmination of Local Model gradients. At this point, the fingerprints can be used to determine the influence each restaker's Local Model gradient had on the updated state of the Global Model. This information can be stored and used later to distribute rewards to restakers based on their contributions.

5. Implement a compensation mechanism that rewards restakers based on the influence of their gradient updates on the Global Model. This can be done by analyzing the impact of each gradient update on the Global Model's performance (e.g., reduction in the loss function) and/or by tracking the usage of the Global Model and attributing a portion of the revenue generated by interaction with the Global Model to each contributing restaker based on their fingerprint.

Disclaimer: I am far from a technical person, as can likely be seen from the contents of this post. I have practically zero background in machine learning, distributed systems, or cryptography. With that said, the feasibility of building the system I propose above should (and I'm sure will) be called into question by those that are more technically minded. The purpose of this post is simply to get the juices flowing!

Cheers and looking forward to some more discussion!