Nimble - The Composable AI Protocol

A Decentralized, Privacy Preserving & Publicly Accessible Artificial Intelligence Internet Infrastructure

Version 1.0

December 19, 2021

Abstract

Massive user data and large-scale artificial intelligence (AI) models are often only accessible to big companies such as Google, Meta and Microsoft. Traditional companies typically enforce strict control over such data, computation and innovation resources. Today's Web3 world is dominated by smart contract solutions supporting very simple computation operations and in lack of generic data access. Thus, only very dummy and simple decentralized applications (DApps) can be built on today's blockchains.

Nimble is a decentralized and composable AI protocol, aiming to leverage data in a privacy preserving manner and provide public large-scale AI models to the general public. It builds the fundamental AI infrastructure for fully decentralized data, model and network ownership. The network achieves this by building a marketplace in which participants — data contributors and model builders — are rewarded to share their data and expertise. The network enables permissionless innovations for developers. Data ownership is given back to users, and artificial intelligence capacities are maintained by the network. Zero knowledge proofs and privacy preserving machine learning build the privacy layer. The network messaging layer is generic to any message types for group messaging, social network comments and image transmissions.

The DApps pay network fees to consume model predictions and validators achieve consensus with *Proof of Gradient* consensus for network security guarantees. More enriched DApps and secure multi-chain Web3 ecosystem can be built on top of it. Such DApps include but not limited to decentralized social network, decentralized social messaging, personalized recommenders, chain agnostic identity system, Web3 reputations, decentralized identities, onchain security protocols, digital asset managements, targeted user growth, decentralized content distribution and beyond. AI models such as wallet activity profiling, asset risks, protocol forensics and fraud detection are learned and served to empower data driven DApps in a privacy preserving and fully decentralized manner.

1 Introduction

The AI powered AI solutions are empowering and securing Web2 companies such as Facebook, Uber, Google and beyond (e.g., real-time data analytics, fraud forensics, wallet and protocol behavior clustering, and rare event detection). Data-driven applications such as ChatGPT, Google, Instagram, and TikTok have been gaining popularity due to many conveniences they provide to their users. However, these services are provided at the expense of user data privacy and centralized content distribution. Decentralizing data-driven applications and bringing them to Web3 can resolve such issues and distribute the wealth from private data to the users.

Nimble is a paradigm change with composable AI capacity. In this way, chatgpt, image generation models, automated blockchain security, risk management, decentralized social networking, Web3 content distribution and personalized recommendation DApps can be easily built to achieve permissionless AI innovations. In the current Web3 ecosystem, security, fraud detection, identities, decentralized social & content, and data driven applications are becoming more and more critical. Yet, complex data aggregations and AI models are not accessible on-chain, due to limited memory and stack restrictions on-chain. As a result, data-driven applications cannot be developed in a decentralized way, and many applications for Web3 still rely on centralized data servers to operate such as NFT marketplaces. With a central server managing data functionalities, it is difficult to achieve the core initiative of Web3, the redistribution of data and network property to the users.

Nimble is to build a composable and decentralized AI protocol. The network consists of overlay network, privacy layer and computation protocol. First, the messaging and overlay network is an autonomous peer-to-peer (P2P) network. Any types of messages can be propagated through the network. It builds the fundamental networks for users and DApps to interact with each other. Zero knowledge and privacy preserving AI technologies ensures user data privacy. The network supports generic data computations such as data aggregations with SQL queries, analytical computations and AI model computations. The protocol will therefore

- build decentralized artificial intelligence solutions with marketplaces for data owners, builders and DApps to exchange values,
- distribute permissionless innovation, data & model ownership back to users with decentralized identities, privacy preserving data repositories and network token rewarded data mining, and
- empower next generation security, fraud detection, identity & reputation protocols, social apps and safer DeFi and crypto asset solutions in a trustless network.

Our development philosophy is governed by principles of *decentralization*, *openness*, and *inclusion*. Decentralization is achieved with an overlay network governing the messaging, payload parsing, and operation processing. The protocol is open to use with publicly published SDKs and APIs. Community members can contribute to the code development and product direction discussions by participating in the DAO.

Section 2 discusses the network basics and Section 3 proposes the AISync protocol for network scalability. The decentralized learning is detailed in Section 4. The consensus protocol is illustrated in Section 5, while network fraudness and communication efficiency solutions are discussed in Section 6 and Section 7. Section 8 and 9 show modular mechanism design and multichain onchain integrations. Applications (Section 10) and related work (Section 11) are discussed before concluding the paper (Section 12).

2 Network Basics

In this section, the design is discussed with a focus on data learning. Data aggregation and analytics can be similarly achieved by replacing model computations with analytics computations. At the same time, we mainly focus the discussions on user private data sharing since public data publishing can be uploaded to the network similarly or in a much easier manner. To achieve the design goals, the following key architecture components formulate the network:

- **Privacy Preserving Gossip Learning.** Data sharing and learning is done so that user privacy is preserved at all times. Users have full rights over how their data is used in the peer-to-peer network. The data is propagated to other nodes anonymously in exchange for token rewards by contributing to the gossip learning protocol. The user data remains fully private with privacy preserving learning. The network is focused on identity, security and personalization models initially, which can be easily extended to other models in the future.

- **Compressed Model Learning.** In decentralized learning, model parameters which can be quite large in size to be propagated across the network. To resolve such bottleneck, the models are compressed via low-rank tensor compression, which reduces redundancy in large-scale tensors and mix-precision representation, by omitting superfluous bits from the model parameters. Both compression methods are known to not cause noticeable model accuracy loss while significantly decreasing the model size.

- **Proof of Gradient Consensus.** Malicious behaviors such as model parameter pollution and faking data could exist in a trustless learning environment. To achieve network robustness, weighted voting with proof-of-stake (PoS) is used as a consensus mechanism for governing the data collection and learning. The weighted voting will allow us to leverage information on the nodes' on-chain behavior patterns in order to help the network reach consensus more safely. In the future, data sampling and model parameter sampling methods can be implemented to further avoid network corruptions.

The key market participants in the network are:

- **Users and Data Nodes.** Data nodes of the network represent users. Data nodes are incentivized by the network token rewards to propagate local data and models around the network in a privacy preserving manner. More specifically, they may participate in on-chain Data Market to broadcast encrypted data and model parameters to other nodes in exchange for Nimble tokens in a trustless manner. They are provers in the verifiable computaion with validators. In the following of the paper, Data Market and Local Model Market are used interchangeably because user data are represented as local models for privacy, and User and Data Node are used interchangeably because from the network perspective users are just data nodes.

- **Validators.** Validators maintain the blockchain through PoS with weighted voting. The nodes who deposit enough stakes in the network's stake pool can be the validators. The network prevents them from acting maliciously during the process by taking away the stakes of faulty nodes. They also act as data aggregation and model learning nodes. They process and learn data in a decentralized way via gossip learning protocol. Validators are verifiers of data node data for decentralization. Meanwhile, the models learned by each validator are verified with predefined evaluation metrics of each model. They have to prove to each

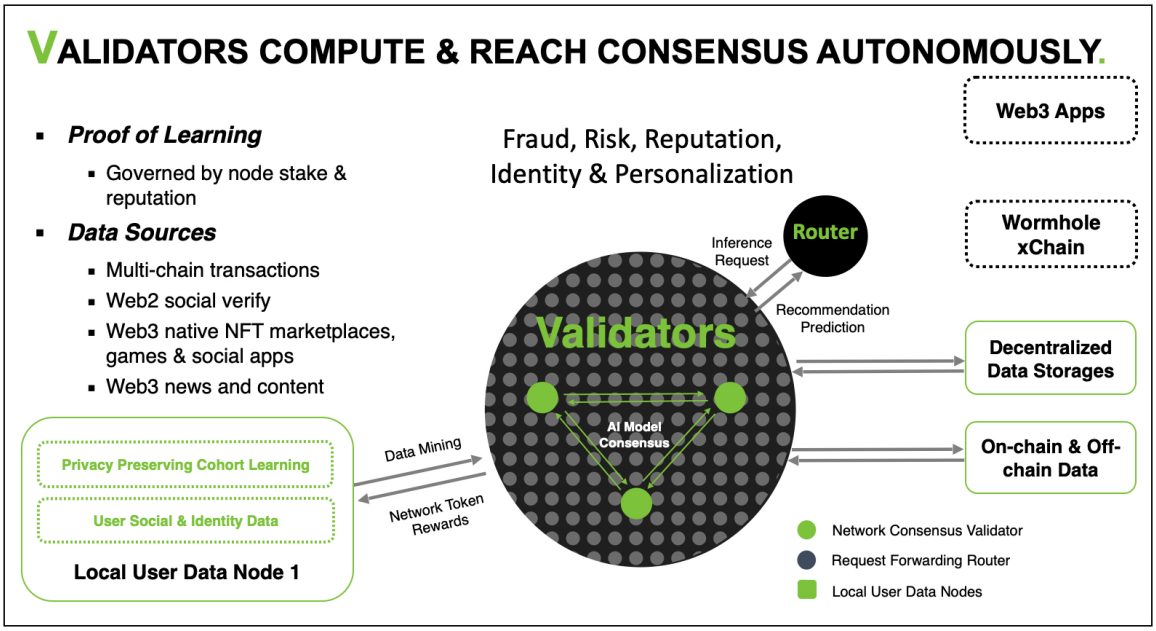

Figure 1: Nimble Decentralized Network Architecture.

other on the model learning results and model evaluation metric improvements. Reputations can be built for efficient validator discovery since they operate autonomously without centralized control to speed up the computations.

- **Clients and Builders.** Clients make data aggregation and model inference queries to the network. Those queries are answered by the validators with suitable data and model in a verifiable and tamper-proof way at the Compuatation Market. The clients who wish to go a step further and build their own model may build models for validators at the Builder Market. Builders can contribute models to the network by earning client Nimble token rewards. To focus the discussions on network operations, for simplicity, we do not differentiate clients and builders in the rest of the paper since they are both consumers of the infrastructure.

- **Routers**: Routers are the nodes who propagate and manage the orders from the clients. They match and verify the orders with some transaction fee incentives. They ensure trustless delivery of information to the clients. Therefore in Nimble, the clients can focus on building their applications. Routers only propagate the requests for Nimble token rewards, while the verifications and model validations are handled by validators.

In data-driven Web2 applications, data collection, storage, and learning is done by one central entity. Such system results in serious data privacy breaches. In order to resolve those issues, we replace the whole process by introducing three decentralized markets coordinated by multiple entities: Data Market (a.k.a., Local Model Market), Computation Market (a.k.a., App Query Market), and Builder Market. The supply and demand for data, models, and queries intersect at these markets. They are verifiable and tamper-proof information markets where the exchanges are made publicly through on-chain orderbooks:

1. Participants add offer (sell) and bid (buy) orders to the orderbook.

2. When a pair gets matched, two parties add a deal order to the orderbook.

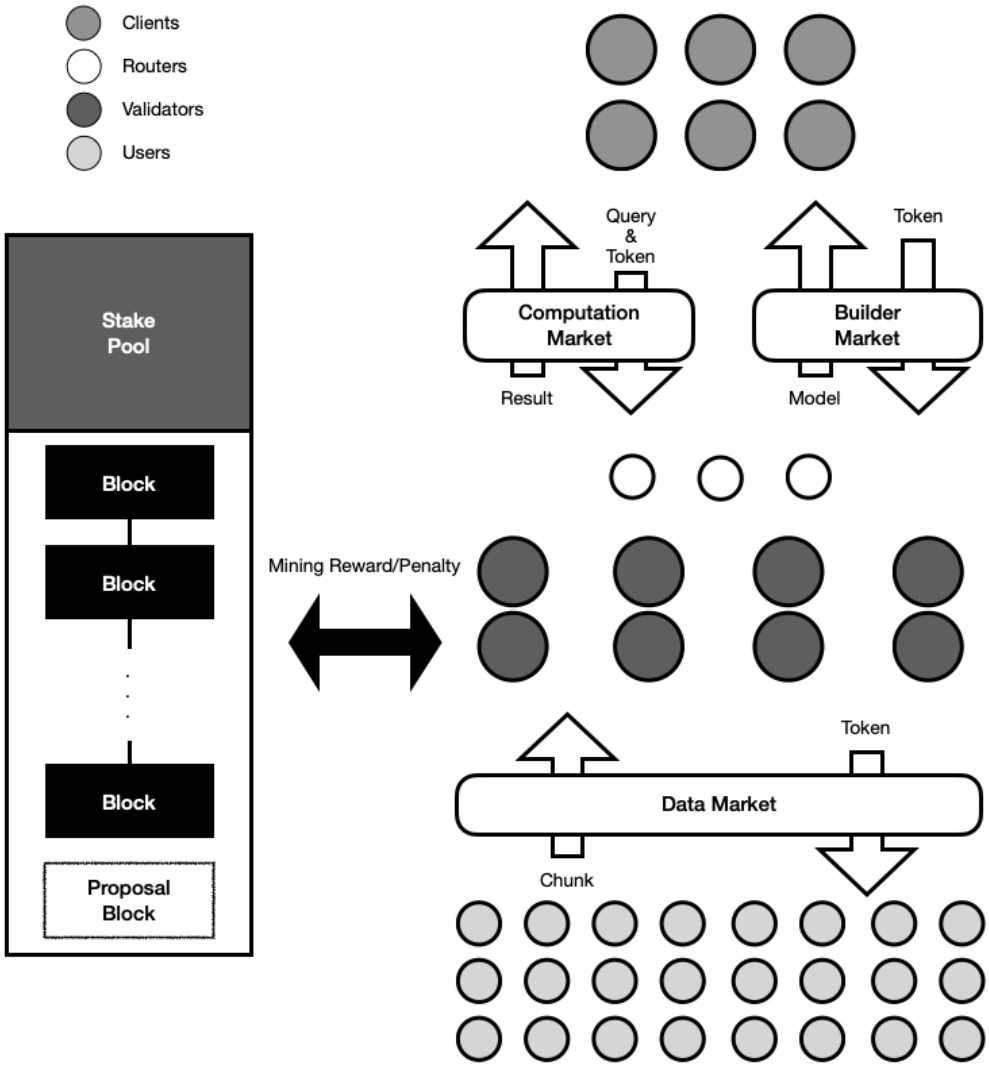

Figure 2: Nimble Cycle.

3. The seller provides proofs for their service, and the proofs get verified.

4. The buyer pays the seller, and the orders get cleared from the orderbook.

The main driving force of the network tokens is the three markets. In these peer-to-peer algorithmic markets, the transactions, orders, and proofs that are involved in the trades should be submitted to the blockchain so that the trades can be verified by the validators. The participants get rewarded for the valid trades only, and this motivates the nodes to not act faulty.

- **Data Market.** The users may participate in the Data Market if they decide to propagate their data pseudo-anonymously in exchange for the token rewards from the validators. They post the offer orders for their local data and model parameters, and once the orders get matched with bid orders, they get passed to the nodes with the matching orders in a privacy-preserving manner. The data and model parameters are backed by verifiable proofs to guarantee their integrity.

- **Computation Market.** There are two types of services provided by the validators to the

clients at the Computation Market: data aggregation and model inference. The Computation Market is a decentralized alternative to the data servers of centralized platforms. It works like a job recruitment market; a client has a query (job) in mind and looks for the validators with data and/or model (caliber) that can answer its query. The routers guarantee that the results provided by the validators are trustworthy.

- **Builder Market.** The model builders contribute their expertise to the network and in turn collect a fraction of the network fees as rewards. The models are standardized with different prediction goals and validators select the best performing AI models, when fulfilling user queries.

As shown in Fig. 2, Nimble blockchain record keeping is done by the validators. A validator may gain stake in Nimble by depositing Nimble tokens into stake pool. Its stake may get taken away if the node is involved in any malicious behaviors in the network such as not properly validating blocks, manipulating data, or faking query results. We also keep track of reputation scores for the validators, which is determined by the nodes' behaviors in the network. For example, a validator who purchased lots of data from Data Market, used them to train a model with high performance, and leveraged the data and the model to provide clients with query results will have a higher reputation score than a validator who did not participate in the markets often.

The blockchain is maintained via Proof of Stake (PoS) with weighted voting. At every epoch, a block proposer is elected pseudo-randomly from the validators in proportion to their stakes and reputation scores. The block proposer needs to create a new block and propose it. The proposal block is added to the network if the majority of the validators agree that the block is valid through a weighted voting, where the weights are determined by the nodes' stakes and reputation scores. Then, the block proposer gets rewarded for mining a new block.

These specialized nodes form a decentralized marketplace of computational resources, offering their expertise and computational power to compete and find optimal solutions for AI requests. For instance, some nodes might specialize in decentralized finance (DeFi) operations, while others focus on non-fungible tokens (NFTs), smart contract interactions, or decentralized storage solutions.

The Miner Network ensures that users receive efficient, cost-effective, and timely results for their mining work, harnessing the collective intelligence of a decentralized community of experts. This layer coordinates and manages the interactions with the Miner Network, ensuring fair competition, transparent decision-making, and efficient resource allocation. The protocol -

- Cooperates with a miner network that leverages optimization algorithms, smart contract scripting, and decentralized finance (DeFi) protocols to competitively find the best miners to execute AI operations.
- Ensures anyone can join the miner network, deploy their own algorithms, and compete for the best AI operations to satisfy their farming goals.
- Generates modular auctions for each operation supported by each miner. Multi-function auctions are generated for common user actions, allowing optimal solutions to be discovered where operations span more than one action.

3 The Dispatching Protocol

Different AI operations including training, prediction and more formulate the mining request pool. Request dispatchers monitor such a pool and route them to the right auctions. Dispatchers

exchange request maps to expand the visibility of the pool. With AI operations specified by the DSL, dispatchers can accurately and efficiently discover the specific auctions. This match-making is analogous to matching riders and drivers by ride-hailing apps.

Gossip protocols and overlay networks are well discussed in the literature on which operations and other network communications are performed [1–6]. Here, the discussion is focused on the *AISync* protocol for application in the protocol. The protocol consists of an operation map, operation propagation, and operation pool.

An operation map is a list of operations together with unique operation IDs. It is an atomic buffer map maintained by each validator. Each validator maintains two operation maps: one for the general purpose opeartions from users and the other for the structured operations in DSL formats. Operation maps are exchanged among validators with the operation propagation algorithm. The algorithm periodically fetches operation maps from its neighbors (i.e., connected validators). It merges new operation maps with the existing ones. The algorithm also periodically discovers new validators as neighbors by removing existing ones with low reputation scores, computed based on network token stakes and the history of mutual interactions. The operation pool is a distributed hash table formed by the operation maps.

The Operation Dispatching Layer is a crucial intermediary between the network users and miners. It plays a multifaceted role in optimizing and safeguarding user operations as they move through the protocol. Its primary responsibility is efficiently managing and organizing these operations as they flow in from the network users. Using event-driven architecture and real-time data batching, this layer ensures that user operations are processed in an orderly fashion. It employs a bundling mechanism to group-related operations, reducing blockchain network congestion and optimizing gas costs. By sequencing and prioritizing such operations, it maintains the consistency and integrity of the blockchain network. This layer acts as the traffic controller and matching protocol, directing operations toward validators and miners for validation and execution, ensuring a streamlined and efficient process.

Additionally, this layer verifies the authenticity and integrity of each operation by checking the associated digital signatures, which must be validated to ensure that the operations originate from authorized users and have not been tampered with during transit. Furthermore, this layer implements business logic rules and access control policies. It enforces constraints and conditions on user operations to prevent unauthorized or malicious actions, ensuring that only valid and secure operations proceed to the miners.

- Positioned as the middleware layer between users and miners, the dispatching layer optimizes and organizes AI operations.
- Consists of bundling operations with specialized optimization algorithms.
- Optimizes and bundles related AI operations for more efficient execution to reduce blockchain network congestion and optimize gas costs.
- Ensures proper sequencing and prioritization of operations to maintain consistency and integrity in the blockchain network.

Privacy Considerations. Dispatching involves two processes: revealing AI operations to miners and revealing miner auction bids to the network. For the initial launch, miners participate in the network with staking. Operations are batched for processing to minimize cheating behaviors (e.g., DeFi operation understanding operations). In later network releases, fully homomorphic encryption (FHE) will be implemented to enable private computations on encrypted data [7].

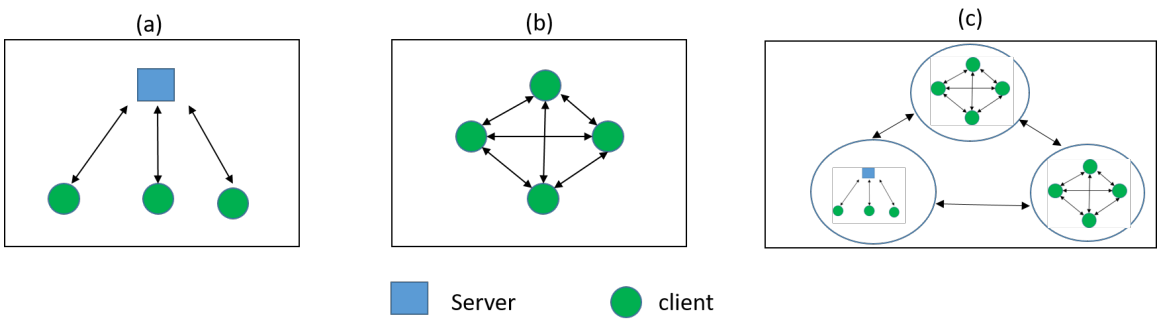

Figure 3: (a) A centralized learning framework. (b) A partially decentralized learning framework. (c) A fully decentralized learning framework.

4 Privacy-Preserving and Decentralized Model Learning

Traditional machine learning techniques require the collection of local data to one centralized data server. In such centralized learning process:

1. Local/personal data are transferred to a centralized server.
2. The server performs learning on a master model.
3. Model inference is controlled by the server.

A Centralized learning framework, shown in Fig. 3 (a), does not have a native mechanism to ensure data privacy and security, and can easily collapse if the server is attacked or corrupted. Moreover, the revenues made from user data is not distributed to the users who actually own them.

In partially decentralized learning approaches such as federated learning, the edge devices/nodes and a central server learn collaboratively without sharing local data as shown in Fig. 3(b). The general process of federated learning is:

1. Central server has a global model and local devices/nodes download the current global model.
2. The local devices/nodes perform learning locally with their own data.
3. The central server collects and summarizes the changes to update the global model.
4. The new version of global model is pushed to every devices/nodes.

Essentially, in decentralized learning, the central server's ability to learn and the need to collect user data are decoupled. In this way, the privacy of user data is preserved. Yet, the overall learning is still governed by one entity, which means that the whole network breaks when the server fails technologically or ethically. Moreover, the ownership of data is still not fully returned to their owners as there is no reward for the local nodes who helped the central server to learn with their local data and computing power.

The key idea behind protecting user data ownership and privacy is to adopt a fully decentralized learning method like gossip learning. In fully decentralized learning, the whole learning is coordinated by the local nodes without a central server orchestrating it as shown in Fig. 3 (c) [8]. The pseudocode for gossip learning is shown in Alg. 1. In this framework,

Algorithm 1 Gossip Learning

$w_k \leftarrow$ init()
loop
  wait(Δ_g) ▷ Wait for a time unit
  $p \leftarrow$ select() ▷ Select a node
  send compress(w_k) to p
  $w_r \leftarrow$ receive() ▷ Receive model from another node
  $w_k \leftarrow$ merge (w_k, w_r)
  $w_k \leftarrow$ update(w_k, D_k)
end loop

1. A node k initializes a local model w_k.
2. Another node p in the network is selected by a sampling service.
3. Model parameter w_k is compressed and sent to p.
4. It receives a model w_r.
5. The node merges it with the local model w_k.
6. Then, updates w_k using the local data set D_k.

Merging is typically achieved by the weighted average of the model parameters. The most common update method is stochastic gradient decent (SGD).

Algorithm 2 Gossip Learning User u

$\theta_u \leftarrow$ init()
loop
  wait(Δ_g) ▷ Wait for a time unit
  θ_u = update(D_u, θ_u) ▷ Update local model
  $k_1, ..., k_n \leftarrow$ selectValidators() ▷ Validators from Data Market
  send encrypt(D_u) to k
  send compress(θ_u) to k
end loop

Algorithm 3 Gossip Learning Validator k

$\theta_k \leftarrow$ init()
loop
  wait(Δ_g) ▷ Wait for a time unit
  $D_r \leftarrow$ receive$(D_{u_1}, ..., D_{u_n})$ ▷ Receive data from users
  $\theta_r \leftarrow$ receive$(\theta_{u_1}, ..., \theta_{u_n})$ ▷ Receive models from users
  $D_k \leftarrow$ merge(D_k, D_r)
  $\theta_k \leftarrow$ merge(θ_k, θ_r)
  $\theta_k \leftarrow$ update(θ_k, D_k)
end loop

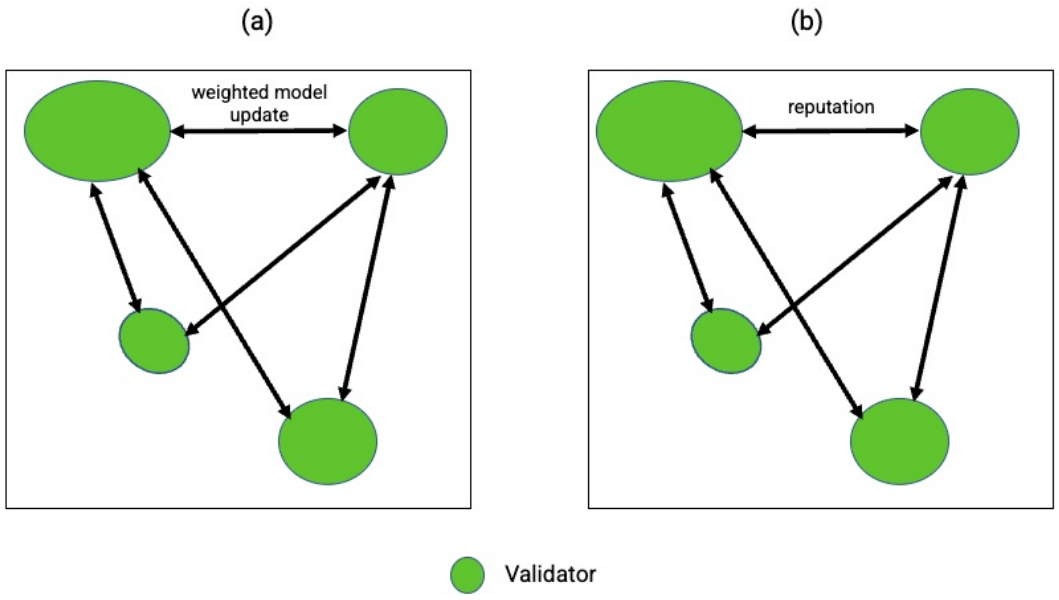

Figure 4: Weighted voting with the Proof-of-Stake.

Our decentralized learning is based on the gossip learning. The users propagate their data and/or model parameters as described in Alg. 2, and the validators take the role of model learning as shown in Alg. 3. In our fully decentralized learning,

1. Users perform local learning with local data.
2. Users offer their data D_u and/or model parameters θ_u in Data Market.
3. Encrypted data and compressed model parameters are transferred to the validators, who won the bid, in a privacy-preserving manner.
4. In return, the users are rewarded with network tokens.
5. The data and/or model parameters from the users are merged with the validators' local data D_k and model parameters θ_k.
6. The validators update local models with the local data and model parameters.
7. Validator nodes with enough stakes and reputation scores validate 1 - 6 and get rewarded with network tokens.

For decentralized learning in a trustless network, we need a consensus mechanism to deal with malicious and dishonest nodes who want to corrupt the learning.

5 Proof of Gradient - Model Learning & Inference Consensus

The PoS with weighted voting is employed as a consensus mechanism in Nimble. Compared to simple majority voting, weighted majority voting is more immune to Sybil attacks where an attacker creates and controls multiple nodes and manipulates the voting. If the voting is weighted by the amount of stake that each validator has, it is much more difficult to execute a Sybil attack

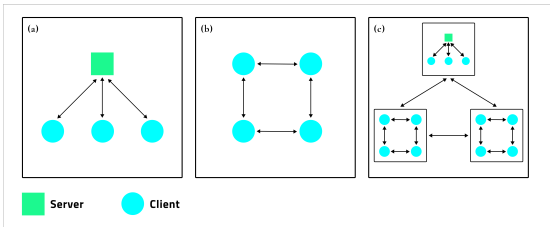

Figure 5: A comparison of centralized and decentralized learning. a) centralized learning, b) decentralized learning, and c) decentralized learning with each validator running one or more network nodes.

because the attacker has to spend significant amount of resources to gain high stake in the network as demonstrated in Fig. 4.

The network robustness comes from the fact that the nodes with bad behaviors such as tampering data and forging model updates and/or inference can be penalized by taking away their stakes, making them suffer economically. To be extra safe, reputation scores, which reflect validators' data storage, model performance metrics, and past interactions with other nodes in the network, will be considered in the weighted voting.

During data learning, the validators performing the learning through the merge and update functions from Alg. 3 need to provide proper proofs that they are not forging the process. Otherwise, their model training will not be approved by the network. Likewise for model inference, the validators need to prove that their query results are valid. Not doing so will cause their stakes and reputation scores to be reduced.

Under this mechanism, the speed of validating a new block will dictate how fast the models learn and predict. Therefore, we may accelerate the voting process by putting a threshold on the stakes and reputation scores for deciding which nodes can participate in the voting.

To participate in network consensus validation, validators must have a minimum amount of staked Nimble tokens. The staked amounts proportionately affect the $2f + 1$ stake weighted *PoAv* during operation dissemination, vote weights, and leader selection during operation recognition and ordering. Validators decide on the split of rewards between themselves and their respective stakers. Stakers can select any number of validators to stake their tokens for a pre-agreed reward split. At the end of every epoch, validators and their respective stakers will receive their rewards via the relevant on-chain modules. Any validator operator with sufficient stake can freely join the Nimble consensus. The network enablement processes can set all parameters, including the minimum stake required.

AI Mining. AI operation requests coming from users and dApps need to be processed by the network for model updates (e.g., operation understanding model, AI routing, and general

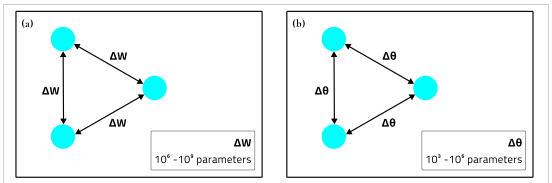

Figure 6: With compressed learning, model parameters can be compressed up to 1,000 times without sacrificing model performances.

purpose operation understanding). Validators are rewarded by improving model performances with new data and serving requests. Users earn network tokens by contributing data. DApps pay network fees for such requests. Early users get more network tokens due to diminishing network token emission rates.

6 Handling Corrupted and Faked Data

Different methods will be implemented to handle corrupted and faked data:

- Corrupted data samples are from valid users, but the data may miss partial information. We want the learning models still work for this kind of corrupted data. Therefore, data corruptions will be considered in the training process via adversarial training or distributionally robust training [9]. The robustness against corrupted data samples can be further improved by implementing a simplified version of the self-healing method [10]. In practice, many data corruptions are unpredictable, so even a model from robust training may still not work well. Instead, we can add some plug-in blocks to the trained model, such that the errors can be identified and mitigated in the prediction process. Specifically, a low-dimensional manifold could be learnt to characterize the "good" behavior of every layer, and the undesired features outside of the manifold can be removed in the forward propagation to avoid or mitigate prediction errors.
- Faked data samples are unwanted in the learning: including them in the learning process can produce a model that provides wrong prediction results. To handle faked data, we will implement a data selection process [11] based on two criteria. Firstly, before the model is retrained, we only select data samples that have low loss function to participate in the training. This will prevent faked data to cause degraded prediction performance. Secondly, we will set an upper bound for the number of samples selected from each group. This will prevent system collapse caused by the infinite faked data produced in the network.

7 Improving Communication Efficiency via Compressed Learning

A fundamental challenge of a decentralized learning framework is how to handle the huge communication cost caused by the large model size. This challenge will be solved efficiently via compressed training [12, 13] in our framework. Specifically, we will implement tensor-compressed training, which has achieved the best compression so far in the training process.

Low-rank Tensor-Compressed Training. As shown in Fig. 7, the model parameters (such as weight matrices and embedding tables) of recommendation systems have lots of redundancy. Because of such redundancy, low-rank approximation can be used to approximate the original model parameters with orders-of-magnitude less model parameters. In the training process, the original 2-D or 3-D model parameters $\mathbf{W}$ will be folded into high-dimensional data arrays (e.g., tensors). Each tensor can be represented with some low-rank factors $\boldsymbol{\theta}$ in a specific compact tensor decomposition format. In the training process, we only need to compute $\boldsymbol{\theta}$ instead of $\mathbf{W}$. As illustrated in Fig. 8, typically $\boldsymbol{\theta}$ has $10^3 \times 10^5$ variables while $\mathbf{W}$ has $10^6 \sim 10^9$ variables, so thousands of times of compression can be achieved on large recommendation system models [12].

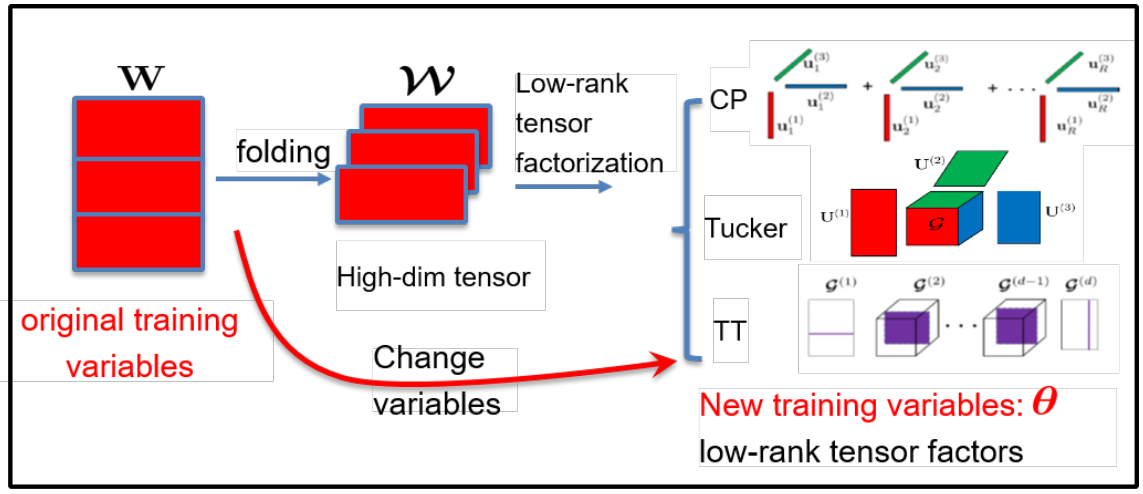

Figure 7: Reducing model parameters by compressing training variables via low-rank tensor factorization.

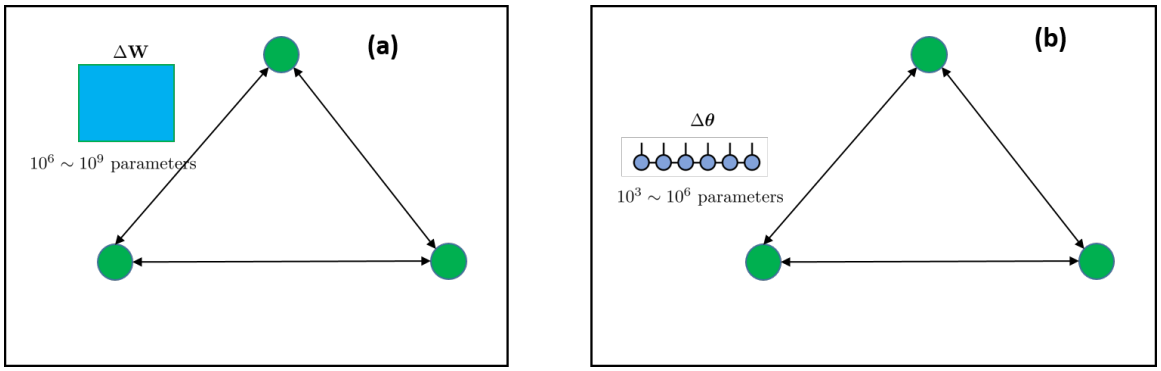

Figure 8: (a) A conventional decentralized learning method needs to share the full-size model information. (b) In our proposed tensor-compressed decentralized learning method, we only share the compressed tensor factors to save the communication cost.

Communication Cost Reduction. To address the communication bottleneck of decentralized learning, only $\boldsymbol{\theta}$ needs to be transferred as shown in Fig. 8, so the latency caused by network communication can also be reduced significantly. Low-precision and mixed-precision representations can also be used to represent the compressed parameters, which can further reduce the communication cost. The key idea is to use a small number of binary bits to represent all variables in the tensor factors. Promising numerical results and hardware demo have been reported in [14, 15]. Such low-precision representation will greatly save the computing, memory and communication cost in federated learning, but it will not cause significant loss of prediction accuracy due to the error-resilient nature of neural networks. The compressed model parameters can be stored on local devices with small memory cost to enable fast inference.

8 Modular Mechanism Design

As discussed in Section 2, the mechanism design consists of the mediator (i.e., the network governed by the Nimble consensus) and the miners, all of which are autonomous agents optimizing for their self-interest. Generic miners may fulfill the entirety of a user's request, while specialized miners may fulfill only a single operation in a highly optimized way. For generality, the rest of this section's discussions focus on specialized miners.

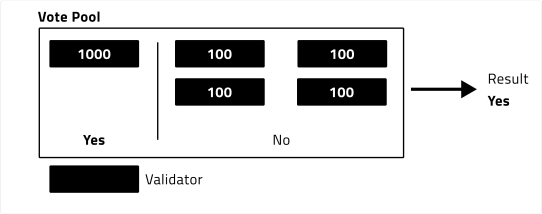

Figure 9: A simple example of weighted voting for model predictions with binary classifications.

Two categories of miners participate in the Nimble protocol. Processors are defined as miners that operate on sub-operations of users' requests or AI training operations. Processors perform operations like onchain executions. Processors unpack operations into multiple transactions and then submit a separate operation for bundling. Bundlers are defined as miners that handle multiple transactions and AI operations like block-building.

In such a multi-agent setting with mediators, processors, and bundlers, an *outcome* is an allocation of operations among processors. Each allocation is sequenced and executed on-chain by bundlers. This becomes a multi-stage mechanism design problem [16–21]. In our modular design, the multi-stage game is modeled as independent second-price sealed-bid auctions (i.e., a special case of Vickrey auctions) for agent bid truthfulness [22, 23]. The settlement market runs a unified auction layer with a modular, composable, and multi-stage design for scalability. In an ideal world, optimizing operation execution is a centralized mechanism design problem. However, the growth of operation categories introduces the notorious dimensionality problem, making the optimization computationally prohibitive. While network traffic remains low, it may be possible to maintain a central auction for miners. As network volume grows, it will become necessary to run modular auctions.

In modular auctions, miners will compete in each child auction for the portion of the user's requests they can fulfill. Generic miners may fulfill the entirety of a user's request, while specialized miners may fulfill only a single operation in a highly optimized way. Modular auctions are created to satisfy atomic AI operations, and composite auctions are created to satisfy operation sets. Through composability and modularity, the protocol can generate optimal solutions from miners without sacrificing performance. In the final stage of the auction, offerings from miners are submitted to Bundlers to finalize the bid. These bundlers are a specialized type of miners that combines offerings into a completed bid. This multi-stage mechanism design is adopted for flexibility, scalability, and extensibility [19].

9 Multi-chain Contracts

The network is accessible by smart contracts and cross-chain through Wormhole Network. Wormhole Network is a decentralized cross chain messaging protocol, supporting most mainstream blockchains. Wormhole Network is not only a token bridge but a generic cross chain messaging layer. Through Wormhole core messaging layer, Nimble communicates with blockchains and publishes network messages and model predictions multi-chain.

User operation classes, execution, dispatching, validation, and error handling are key components of the operation contract. We focus our discussions on the operation class, execution, and dispatching since validation and error handling come naturally. The AIOperations class defines operations through the taxonomy. It also consists of important operation utility functions. The OperationDispatcher is the unified interface for solvers to interact with, and it routes operations

to specialized OperationExecutor. OperationExecutor class handles particular operation operation executions. Operations are validated before being sent to the OperationDispatcher. ErrorHandler is called by different components whenever errors are detected. The flow is illustrated in Fig. **??**.

10 Applications

Nimble is to enable the next generation DApps. Ecosystem applications include but not limited to:

- Wallet and protocol forensics for identity, security and fraud detection.
- Intent protocol design with large action model for onchain operation mapping.
- "Smarter" protocols such as more complex autonomous market makers (AMMs) and lending protocols that are driven by real-time data about wallets, users and contracts.
- Safer protocols with fraud, anomaly, and network intrusion detection.
- Data-driven DeFi and NFT trading.
- Web3 protocols that require user data such as identity, reputation, social behavior, and recommendation protocols.
- Web3 native social networks like LinkedIn, Facebook, TikTok, Instagram and so on while preserving user data privacy and ownership.
- Data-driven investment, grants, and social decentralized autonomous organizations (DAOs) that run based on data analytics.
- Chain agnostic identity system leveraging offchain and onchain user data.

Case Study - Decentralized Data Aggregation. Data engineers can contribute to the network as miners as well. Data aggregation is the process where raw data collected from multiple sources are processed and presented in summary statistics such as average, sum, and count for statistical analysis. For example, we can gather information about NFT transactions for a certain period of time and provide observations such as:

- Average, minimum, and maximum price in 24 hours.
- The average frequency of transaction by NFT.
- The number of buyers by NFT collection.

In other words, a data aggregation tool enable its users to ask questions (queries) and get answers (responses) from large amount of data.

The most widely used programming language for data management is Structured Query Language (SQL). SQL can be used to query, manipulate (insert, update, and delete), define, and access data. Its keywords such as GROUP BY, MIN, MAX, COUNT, AVG, and SUM let us easily get summarized statistics from raw data.

As illustrated in Fig. 1, DApps built on top of Nimble can use usual SQL queries to get the onchain data analytics with some transaction fee. Once a query is made by the application, the query will be posted to the Computation Market orderbook, and the routers may propagate the order

around the network. Then, the validators with proper data will process the data and respond to the query. Usually a central database is involved in many decentralized data solutions due to the low speed of blockchains. In Nimble, we can use varying stake and reputation score thresholds to speed up the data aggregation process. In this way, Nimble can respond to large enterprise scale data requests without moving the data off-chain.

Moreover, Nimble supports extract, transform, and load (ETL) pipeline. Both batch ETL, where data is accumulated and processed in batches during a batch window, and online ETL, where data is processed as soon as it arrives, are supported. Batch ETL is useful for dApps that want to be cost-effective and need to process large volumes of data. On the other hand, online ETL can help dApps that have delay sensitive functionalities such as fraud detection. Since Nimble has both batch and online ETL functionalities, it can be used for both online analytical processing (OLAP) and online transaction processing (OLTP).

As mentioned above, Nimble data consensus mechanism is PoS with weighted voting. As in model learning and inference, the validators participate in verifying data collection and processing on-chain. A node that performs data aggregation needs to generate and submit the proofs. Failing to do so will result in losing their stakes. In addition, stake and reputation score threshold is actively managed in order to respond to different types of queries made by dApps. For the queries that are complex and require lots of data, we will lower the entry threshold so that many validators can participate. On the other hand, the queries that are simple and focus on short atomic data can be managed by fewer validators by raising the threshold.

11 Related Work

This section discusses the evolution of decentralization technologies, natural language deep learning techniques, and existing decentralized AI solutions.

11.1 Bitcoin

In the evolution of blockchain, Bitcoin was the first viable solution [24]. Bitcoin is a decentralized digital currency that can be transferred on the Bitcoin network. The peer-to-peer network of nodes running Bitcoin software maintains the blockchain, which is a public record of bitcoin transactions. The network nodes independently store a copy of the blockchain and validate transactions, appending them to their own copies of the blockchain and transmitting the additions to other nodes. In every epoch, a new block that contains a new batch of valid transaction records is added to the blockchain and issued to all nodes. The miner nodes do the record-keeping of the blockchain by using their processing power. The easy-to-verify but hard-to-solve proof-of-work (PoW) is required in each new block to be accepted by the rest of the network. Merkel Trees are adopted so non-miners can use the block header to verify a new blockchain without checking all past transactions. Once a new block is accepted by the rest of the network, the miner who successfully solved the problem is rewarded with newly created bitcoins and transaction fees.

11.2 Ethereum

After Bitcoin, numerous additional blockchains were built. Among them, Ethereum pioneered the smart contract platform [25]. Ethereum inherits the Bitcoin ecosystem's consensus, decentralization, and cryptography. While Bitcoin is limited as a simple ledger, the Ethereum blockchain has a built-in Turing-complete programming language that allows anyone to write smart contracts and

develop decentralized applications (dApps) on top of it. Ether (ETH) is the native cryptocurrency of the Ethereum blockchain that is paid to the miners as transaction fees. Due to their autonomous nature under their own logic, Ethereum contracts are often called smart contracts. The code in the contracts is written in Ethereum virtual machine (EVM) code, a low-level, stack-based bytecode language. The EVM is the runtime environment of Ethereum that includes a stack, memory, gas balance, program counter, and persistent storage for all accounts. When a transaction calls a contract's function, the EVM translates and executes the contract's code into stack operations. A transaction sender must pay the miner, who is running the EVM and adding the transaction to the blockchain, a certain amount of gas fee in ETH. The gas fee system can reduce the amount of spam transactions.

11.3 Proof-of-Stake

Initially proposed by PeerCoin [26], the creation of a scalable Web3 smart contract platform has recently become a focus for the community. Proof-of-Stake (PoS) is more scalable and better suited for general-purpose blockchains like Ethereum. Applications like decentralized finance (DeFi), non-fungible tokens (NFTs), minting, and sales all require high transaction volumes and benefit from improved scalability. In PoS, a blockchain employs a network of validators who stake their own tokens into a pool to get a chance to validate new transactions. If a validator is chosen to update the blockchain, they receive the reward. A majority consensus has to be reached for a block to be accepted and added to the blockchain. Weighted voting can be adopted to improve the robustness of the consensus, where the weight is determined by the node's profile, such as the amount of stake and reputation.

11.4 Fully Decentralized Learning

In contrast to traditional machine learning techniques where all local data stored at local nodes are uploaded to a centralized server, decentralized learning like federated learning and gossip learning enables the client data to be only kept on local devices/nodes without being shared in the network. Generally in federated learning, local models are trained with local data and the model parameters are exchanged at some frequency to learn a global model that is shared by all nodes. Under such set up, a central server orchestrates the learning process. The central server initializes a global model and broadcasts it to the network and then individual nodes will train the model with local data. Once the local training is complete, the central server collects the training results from the nodes and updates the global model with some rule (generally averaging) which is again transferred to every nodes. On the other hand, in fully decentralized learning, such as gossip learning, there is no central server and the nodes themselves coordinate the learning.

11.5 Low-rank Tensor Compressed Training

A tensor is a multi-dimensional collection of numbers that can be thought of as a matrix extended to higher dimensions than 2-D. The motivation behind tensor decomposition comes from matrix factor analysis, which breaks down a matrix into two factors that contain latent relationships. Low-rank tensor decomposition can compress a large tensor while retaining the latent information inside it. Therefore, it has been one of the most effective approaches to reduce the memory and the computational costs of neural networks. In general, the model parameters are compressed after the training to reduce storage requirement and shorten inference time. However such methods

cannot benefit from the reduced memory during training. Instead by using an end-to-end low-rank tensor compressed training algorithm, the advantage of compression can be enjoyed during training time, and the model can be optimized for the compressed setting.

11.6 Mixed-precision Training

Mixed-precision utilizes numerical formats with lower precision than 32-bit floating point which is a mainstream numerical format used to represent model parameters in machine learning. Lower precision numbers translate to less memory and faster mathematical operations. As a result models trained with mixed-precision training algorithms require less memory to be stored, and their training and inference time gets shortened. Such benefits come without losing any task-specific accuracy compared to the full precision training by identifying the parameters that require full precision (i.e., 32-bit floating point) and adopting lower precision representation to the parameters that can be represented with lower precision.

11.7 Personalized Recommendation

Deep learning is widely adopted to achieve personalized recommendations for search, social apps, and user interest discovery. In particular, the state of art is to build a user side deep learning model (a.k.a., user tower) and an item side deep learning model (a.k.a, item tower), together with a ranking model to combine them for final predictions [27–29]. The model parameter size can be hundreds of Gigabytes or even Terabytes. Recommendation models are the key driver and brain of the mainstream Web2 applications. A Web3 data intelligence infrastructure is needed to support such model computations and real-time predictions to indeed decentralize data ownerships.

11.8 Identity, Targeted User Growth, Fraud Detection and Risk Management

Together with personalized recommendation, identity, fraud detection and risk management form the key data intelligence solutions powering the current Web2 data and social applications. The blockchain and crypto data are nothing different from Web2 with a) onchain transaction data similar to credit card transaction data in Uber, Doordash, Airbnb and Instacart, b) offchain data including image (e.g., NFTs), natrual languages, and metadata, and c) biometrics data including wallet fingerprints, location, and account verifications [30–32]. A Web3 data infrastructure is missing to get access to such data and build similar technologies for a more secure, identifiable and privacy preserving Web3 world.

11.9 Web3 Analytics, Security LoopHoles and SocialFi

While delivering a lot of value to the world with censorship resistant, Web3 identity and social solutions, the Web3 community is still building very naive solutions from the data intelligence perspective. For example, the mainstream analytics tools (e.g., Nansen) only support very simple analysis, which are solutions long deprecated by Web2 data community. Most of security, fraud, and financial crime issues can be easily detected with intelligent solutions like machine learning and data clustering by using public data only. With emergence of Web3 social, metaverse and games, how to leverage Web2 data intelligence technologies such as personalization, fraud detection, identity and risk management is critical to their success and user data decentralization.

11.10 Decentralized AI Networks

There are numerous protocols being claimed as decentralized AI networks (e.g., Bittensor, Render, SingularityNet, Fetch.AI etc.). Bittensor has a token distrubiton network to miners. It is not decentralized with governance controlled by the team behind the scene. The miners cannot run distributed model training collectively. That is, there are no way for miners to collaborate and thus no permissionless collaborations among data contributors, AI engineers and compute resources. Render, SingularityNet and Fetch.AI build nothing but the minted tokens on public blockchains with a decentralized AI narrative. The space is still in urgent need of a real decentralized AI protocol which is composable and enables permissionless innovations for developers.

11.11 LLMs

Large language model is well studied in Web2 for chatbot, personalized search, recommendation, and user growth applications. For example, one mainstream application is to adopt large-scale AI models and knowledge graphs for named entity tagging and similarity search [27–29, 33–35]. For example, search queries are processed with spelling corrections, query segmentation into semantic units, and entity tagging for such units with human-understandable concepts like place names, companies, industries, etc. Such concepts (i.e., entities) are mapped into a graph to represent their relationships. This lays the foundation for a structural understanding of intents. Another popular approach is representing user queries as embeddings (i.e., a vector of bits). The intents of the queries and items (i.e., products in e-commerce, articles in search, and posts in recommendations) are measured by cosine similarity among them. That is, queries with similar intents and items on similar topics are closer in the mapped space.

12 Conclusion

In this paper, we have proposed an ultimate AI protocol that is decentralized, composable and with data learning and aggregation capacity. Many Web3 solutions out there are not as decentralized as they seem. This is mainly because most popular blockchain applications still use the centralized server system to store user data and provide services around them. On the other hand, the Nimble deals with user data in a fully decentralized manner and can serve the requests made by applications without relying on a centralized backend server. In addition, the transaction fees collected during model inference and data aggregation are naturally circulated around the network, not only to the validators but also to normal users, through the market system. In this way, Nimble can be a solution to returning data, permissionless innovation, and network ownership to the users. Nimble is also client-friendly as we may adjust the the stake and reputation score thresholds for the weighted voting with PoS to speed up the serving time for the requests made by the clients. Therefore, Web3 native data-driven applications can be built on top of it in a truly decentralized way.

References

[1] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Peer-to-Peer Systems II: Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003. Revised Papers 2*. Springer, 2003, pp. 98–107.

[2] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "Topbt: A topology-aware and infrastructure-independent bittorrent client," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[3] L. Zhao, J.-G. Luo, M. Zhang, W.-J. Fu, J. Luo, Y.-F. Zhang, and S.-Q. Yang, "Gridmedia: A practical peer-to-peer based live video streaming system," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*. IEEE, 2005, pp. 1–4.

[4] T. Small, B. Liang, and B. Li, "Scaling laws and tradeoffs in peer-to-peer live multimedia streaming," in *Proceedings of the 14th ACM international conference on Multimedia*, 2006, pp. 539–548.

[5] F. F. E. Dabek, "A distributed hash table," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[6] X. Jin and Y.-K. Kwok, "Network aware p2p multimedia streaming: Capacity or locality?" in *2011 IEEE International Conference on Peer-to-Peer Computing*. IEEE, 2011, pp. 54–63.

[7] D. D. T. F. P. I. M. J. D. R. N. S. Morten DAhl, Clement Danjou and L. T. Thibault. (2023) fhevm - confidential evm smart contracts with fully homomorphic encryption. [Online]. Available: https://github.com/zama-ai/fhevm/blob/main/fhevm-whitepaper.pdf

[8] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[9] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[10] Z. Chen, Q. Li, and Z. Zhang, "Self-healing robust neural networks via closed-loop control," *arXiv preprint arXiv:2206.12963*, 2022.

[11] Y. Roh, K. Lee, S. Whang, and C. Suh, "Sample selection for fair and robust training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 815–827, 2021.

[12] C. Hawkins, X. Liu, and Z. Zhang, "Towards compact neural networks via end-to-end training: A bayesian tensor approach with automatic rank determination," *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 1, pp. 46–71, 2022.

[13] C. Hawkins and Z. Zhang, "Bayesian tensorized neural networks with automatic rank selection," *Neurocomputing*, vol. 453, pp. 172–180, 2021.

[14] Z. Yang, J. Shan, and Z. Zhang, "Hardware-efficient mixed-precision CP tensor decomposition," *arXiv preprint arXiv:2209.04003*, 2022.

[15] K. Zhang, C. Hawkins, X. Zhang, C. Hao, and Z. Zhang, "On-FPGA training with ultra memory reduction: A low-precision tensor method," *arXiv preprint arXiv:2104.03420*, 2021.

[16] R. B. Myerson, "Multistage games with communication," *Econometrica: Journal of the Econometric Society*, pp. 323–358, 1986.

[17] R. Jurca and B. Faltings, "Mechanisms for making crowds truthful," *Journal of Artificial Intelligence Research*, vol. 34, pp. 209–253, 2009.

[18] M. Curry, V. Thoma, D. Chakrabarti, S. M. McAleer, C. Kroer, T. Sandholm, N. He, and S. Seuken, "Automated design of affine maximizer mechanisms in dynamic settings," in *Sixteenth European Workshop on Reinforcement Learning*, 2023.

[19] T. W. Sandholm, V. Conitzer, and C. Boutilier, "Automated design of multistage mechanisms," 2007.

[20] M. T. Hajiaghayi, R. Kleinberg, and T. Sandholm, "Automated online mechanism design and prophet inequalities," in *AAAI*, vol. 7, 2007, pp. 58–65.

[21] J. Gan, R. Majumdar, G. Radanovic, and A. Singla, "Sequential decision making with information asymmetry," in *33rd International Conference on Concurrency Theory*. Schloss Dagstuhl, 2022, pp. 1–18.

[22] J.-J. Laffont and E. Maskin, "Optimal reservation price in the vickrey auction," *Economics Letters*, vol. 6, no. 4, pp. 309–313, 1980.

[23] L. M. Ausubel *et al.*, "A generalized vickrey auction," *Econo0 metrica*, 1999.

[24] S. Nakamoto, "Bitcoin whitepaper," *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*, 2008.

[25] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[26] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, no. 1, 2012.

[27] M. Naumov, D. Mudigere, H. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C. Wu, A. G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy, "Deep learning recommendation model for personalization and recommendation systems," *CoRR*, vol. abs/1906.00091, 2019. [Online]. Available: https://arxiv.org/abs/1906.00091

[28] U. Gupta, X. Wang, M. Naumov, C. Wu, B. Reagen, D. Brooks, B. Cottel, K. M. Hazelwood, B. Jia, H. S. Lee, A. Malevich, D. Mudigere, M. Smelyanskiy, L. Xiong, and X. Zhang, "The architectural implications of facebook's dnn-based personalized recommendation," *CoRR*, vol. abs/1906.03109, 2019. [Online]. Available: https://arxiv.org/abs/1906.03109

[29] H. M. Shi, D. Mudigere, M. Naumov, and J. Yang, "Compositional embeddings using complementary partitions for memory-efficient recommendation systems," *CoRR*, vol. abs/1909.02107, 2019. [Online]. Available: https://arxiv.org/abs/1909.02107

[30] A. S. A. J. Xinyu Hu, Chengliang Yang and P. Molino, "Fraud detection: Using relational graph learning to detect collusion," *Uber Engineering Blog*, 2021. [Online]. Available: https://www.uber.com/blog/fraud-detection

[31] T. Y. E. N. Sergey Zelvenskiy, Garvit Harisinghani and R. Wei, "Project radar: Intelligent early fraud detection system with humans in the loop," *Uber Engineering Blog*, 2022. [Online]. Available: https://www.uber.com/blog/project-radar-intelligent-early-fraud-detection

[32] Y. Diao, "https://www.uber.com/blog/mastermind," *Uber Engineering Blog*, 2017. [Online]. Available: https://www.uber.com/blog/mastermind

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[34] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang, "Gpt-ner: Named entity recognition via large language models," *arXiv preprint arXiv:2304.10428*, 2023.

[35] X. Zou, "A survey on application of knowledge graph," in *Journal of Physics: Conference Series*, vol. 1487, no. 1. IOP Publishing, 2020, p. 012016.

