ML on Chain: The Case and Taxonomy of Machine Learning on Blockchain

Binbin Gu
Department of Computer Science
University of California, Irvine
Irvine, USA
binbing@uci.edu

Abhishek Singh
Department of Computer Science
University of California, Irvine
Irvine, USA
abhishas@uci.edu

Yinan Zhou
Department of Computer Science
University of California, Irvine
Irvine, USA
yinanz17@uci.edu

Juncheng Fang
Department of Computer Science
University of California, Irvine
Irvine, USA
junchf1@uci.edu

Faisal Nawab
Department of Computer Science
University of California, Irvine
Irvine, USA
nawabf@uci.edu

***Abstract*—Blockchain has recently amassed a lot of interest from researchers and practitioners. This is due to its ability to manage data in a decentralized, transparent and accountable manner. However, blockchain applications are limited in terms of compute and cost effectiveness. These limits led to smart contracts—programs that are processed by the blockchain network—being relatively simple, not taking advantage of complex computation methods. A notable example of this is the support of machine learning (ML) to provide data-driven insights in decentralized applications (DApps). Utilizing ML models within smart contracts has been disregarded because ML involves expensive computation. Nevertheless, ML on blockchain is becoming more feasible due to advances in consensus protocols, layer-2, and off-chain solutions. This motivates us to rethink the question of utilizing ML on blockchain.**

In this paper, we shed light on the design space of using machine learning in blockchain applications (we call this *ML on Chain* for short). Although a lot of work discusses the intersection of blockchain and ML, there is little clarity in mapping, understanding, and evaluating the various approaches in this space. This paper contributes to this new area by the following: (1) we introduce a taxonomy of ML on Chain. This taxonomy considers existing and future potential methods in this area and groups them based on their design characteristics. We consider in the taxonomy five off-chain approaches and a baseline on-chain approach. (2) we perform an extensive experimental evaluation of ML on Chain methods. We compare the different 5 groups of solutions across different settings and three ML model types: logistic regression, k-nearest neighbors and neural networks. (3) Using the taxonomy and experimental study, we provide insights about the current state and challenges of the ML on Chain space; we use this newfound understanding to discuss potential future approaches and provide suggestions to address ML on Chain challenges.

I. Introduction

There has been a growing interest in blockchain in the last few years. It has the ability to manage data in a decentralized, secure, transparent and accountable manner. With the support of smart contracts (i.e., programs running on blockchain), DApps (Decentralized Applications) and DAOs (Decentralized Autonomous Organizations) are developed to present new solutions for Internet applications. This has been explored in many areas such as decentralized finance (DeFi) [1], gaming and metaverses [2], [3], and supply-chain [4], where hundreds of thousands of users and hundreds of millions of dollars were amassed for these applications [5]. For example, BitDAO [6] manages more than 2.5 billion dollars for economic alignment, governance, treasury management, and organization.

Machine learning on blockchain (ML on Chain) has immense benefits such as immutability, accountability and transparency of the ML models [7], [8]. In terms of training, ML on Chain ensures that the training process was not tampered with. In terms of prediction (also called inference), ML on Chain ensures that the prediction was on the intended model without malicious interference. However, smart contracts are limited. Training ML models in smart contracts requires paying high monetary fees (e.g., gas on Ethereum). These fees are higher for more complex computations. For instance, doing a scalar product over two 1000-dimension vectors could cost more than $10 on Ethereum. This leads to disregarding a lot of computing technologies that require intensive computation, such as ML on Chain.

Recent advances in blockchain have led to ML on Chain becoming more feasible due to emerging off-chain technologies—i.e., technologies that utilize nodes outside of the blockchain network to perform computation and storage on behalf of smart contracts. This motivates us to rethink the feasibility of ML on Chain. The key challenge in utilizing off-chain nodes is that these nodes can be malicious. Solutions to ensure that off-chain processing is performed correctly include the use of authenticated data structures [9], [10], [11], verifiable computing [12], [13], [14], trusted hardware [15], [16], [17], and distributed byzantine agreement [18], [19]. Although these approaches were explored for various use cases [20], [21], [22], there is little work on showing how to utilize them for ML on Chain as well as little work on understanding their properties and how they compare with each other when used for ML on Chain. We aim to fill this gap in this paper.

In this paper, we provide an analysis and experimental study of the ML on Chain space. In terms of analysis, we propose a taxonomy of ML on Chain that groups solutions into a baseline on-chain approach and five off-chain approaches (i.e., enclave-based, verifiable computing, incentive-based, voting-based, and distributed byzantine ML). In the analysis, we discuss the design implications of each approach on factors such as ML model performance, trustworthiness of training and predictions, and data privacy. Then, we utilize this taxonomy to guide our experimental study where we evaluate and compare the five off-chain and baseline on-chain approaches. In the evaluations, we focus on the practical implications of ML on Chain solutions such as their throughput, latency, and monetary cost. To cover a wide-range of ML approaches, we perform our experiments with three ML models that have different characteristics: logistic regression, k-nearest neighbors and neural networks.

To the best of our knowledge, this is the first work to conduct both a design and quantitative analysis of ML on Chain.The contributions of this paper are:

- We propose a taxonomy of ML on Chain approaches. We use this taxonomy to drill down in each category and analyze various ML on Chain solutions and their design trade-offs. This includes analyzing design and performance characteristics, trust assumptions, privacy and security, monetary cost, and limitations.
- We conduct an experimental evaluation to study and compare ML on Chain solutions. The evaluation is performed on Ethereum with three popular ML models.
- In addition to providing a taxonomy of prior work related to ML on Chain, we foresee new ML on Chain approaches that have not been explored in prior literature or industry systems and include them in the taxonomy and evaluation.

The rest of the paper is organized as follows: We first present background information in Section 2. Then, we propose the taxonomy in Section 3 followed by a more detailed analysis of off-chain methods in Section 4. In Section 5, we show our experiments. In Section 6, we describe the related work and conclude with a discussion of future directions and challenges in Section 7.

II. Background

In this section, we introduce preliminary and background information about ML, blockchain and the integration of ML and blockchain.

Machine learning. In the last decade, there was a lot of interest in applying ML to problems in various domains [23], [24]. This is because of the powerful nature of ML in learning insights from data and applying these insights for future tasks. A ML workflow, as shown in Figure 1, can be divided into two parts: model generation and model use. Model generation mainly contains two components: data preprocessing and model training. Data preprocessing transforms raw data into ML-compatible representations which may include feature selection, data regularization, and normalization. The **model training** refers to the process of generating a *ML model* from an input dataset. Model use is often called **inference** which refers to the process of using the ML model for tasks such as prediction and classification. An epoch of training is when all the training data is used at once and is defined as the total number of iterations of all the training data in one cycle for training the ML model. There are different types of ML models such as neural networks [25], decision trees [26], naive Bayesian [27] and k-NN [28]. Each model type offers different characteristics in terms of complexity, performance efficiency, and accuracy.

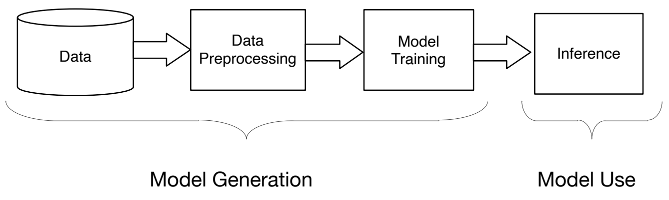

Fig. 1: A simple ML workflow.

Blockchain and smart contracts. Blockchain is an immutable sequence of data records that are cryptographically linked together and maintained by a decentralized network of nodes, called blockchain nodes [29]. Blockchain nodes use a consensus algorithm to agree on the order and content of the data records in blockchain. Consensus protocols can be permissionless, where nodes can join and leave the blockchain network arbitrarily (open membership) [30]. Examples of permissionless consensus protocols are Proof-of-Work (PoW) [31] and Proof-of-Stake (PoS) [32] used by blockchains such as bitcoin and Ethereum. On the other hand, closed-membership blockchains such as multi-organization blockchain consortiums [33] can utilize permissioned consensus protocols such as byzantine agreement protocols [34].

A *smart contract* is a program (i.e., set of functions and state such as memory and variables) that is maintained on blockchain. Smart contracts are stored in the blockchain, and are automatically processed when a transaction invokes a smart contract procedure. After each execution of the smart contract, its state can be updated on the blockchain for checking. Due to their safety, security and trustworthiness, smart contracts are the key components in implementing DApps and DAOs. Deploying and running a smart contract on blockchain often require paying fees (e.g. users need to pay gas fees for deploying and running smart contracts on Ethereum). These fees are high when complex computation is performed or a large amount of data is utilized.

Integration of ML and blockchain. The integration of ML and blockchain has been explored in various contexts. This

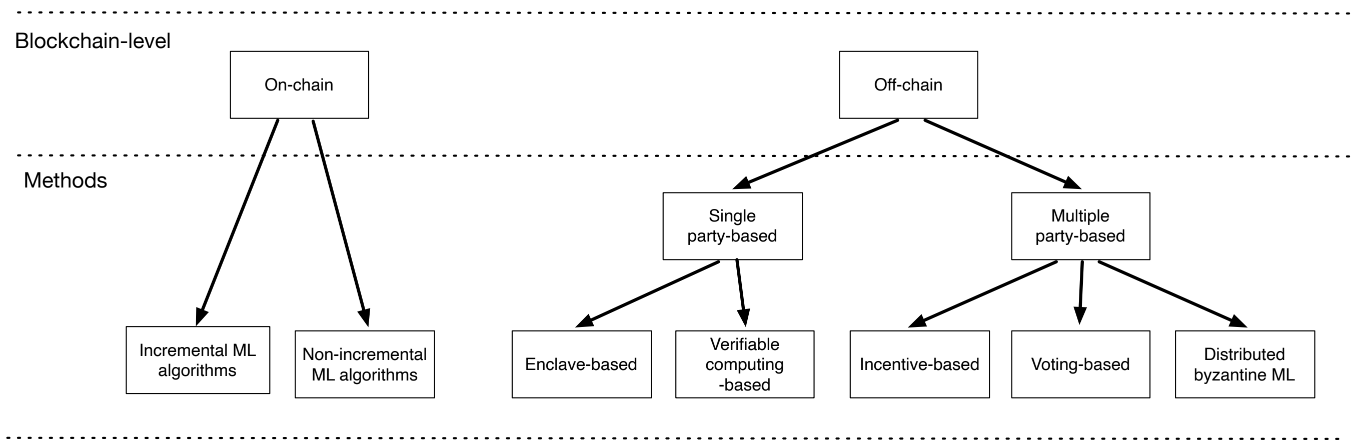

Fig. 2: Taxonomy for ML on Chain.

includes optimizing DApp operation and functionality [35], [36] and utilizing models to gain insights from collected data [37], [7]. For example, machine learning algorithms can be used to identify fraudulent transactions on a blockchain and prevent them from being processed. Involving blockchain in the process of training and prediction enables DApps to utilize ML models without threatening their integrity as trusted, decentralized applications. In addition to DApps, ML use cases that benefit from performing ML on Chain are ones that require features such as transparency, decentralization, privacy, and accountability. For example, federated learning use cases require many of these features.

Off-chain scaling. One of the scaling solutions of blockchain smart contracts is the utilization of off-chain computation and storage. Utilizing off-chain nodes for computation and storage reduces the monetary and performance overhead of performing actions on-chain [38], [39], [40]. The main challenge of off-chain approaches is the security risks of utilizing nodes that are outside of the blockchain network and are thus not governed by the same security guarantees. For this reason, many techniques were proposed to ensure that off-chain nodes will not act maliciously. We present and utilize some of these techniques when we review off-chain methods for ML on Chain.

III. TAXONOMY

A. Overview

ML models can be trained and utilized to do inference on-chain (as part of the smart contract logic) [41] or off-chain (using nodes outside of the blockchain network) [22], [13], [39]. In off-chain training and inference, the on-chain smart contract plays the role of a trusted party that validates and maintains information about the off-chain node(s). This can take many forms that we discuss throughout the paper. As an example, one type of off-chain solutions performs model training off-chain and then submits the model and a zero-knowledge proof of the model's computation to the smart contract to verify.

We propose a taxonomy of ML on Chain that groups solutions based on the following: (1) is the task performed on-chain or off-chain, and (2) is it a model training task or an inference task. In the rest of this section, we discuss the taxonomy groups in more details, and in the following section we provide an analysis of the properties of off-chain methods (Section IV).

B. On-chain ML Training and Inference

In this section, we discuss on-chain training followed by on-chain inference.

On-chain training. The main benefit of on-chain training is the transparency of the training process. We can make sure that the training process was not tampered with and each user on blockchain can verify whether a ML model is generated according to the algorithm in the smart contract. There are several challenges that are associated with running ML model on-chain. This includes the high fees involved in processing and storing the model as well as the latency overhead of writing to blockchain. We study the cost of training different ML models on-chain in the evaluation section (Section V).

In addition, there is a fundamental challenge that faces on-chain ML solutions that is due to the bound on how much computation and data can be processed on-chain within one block. To overcome this challenge, models can be trained incrementally. Specifically, the dataset is split into smaller datasets so that each part can be performed in a separate block. After all the parts are processed, the process terminates.

Figure 3 illustrates how incremental training works with multiple blocks on blockchain. Each block updates the ML model based on a subset (shard) of the whole dataset. Once the model is updated based on the current shard, it will be passed to the next block to continue training. An epoch of the training process is complete when all the shards are processed once. Once an epoch of training is performed, the process repeats for the next training epoch.This training process terminates after a specified number of epochs are performed. One limitation of this incremental learning approach is that the shard size

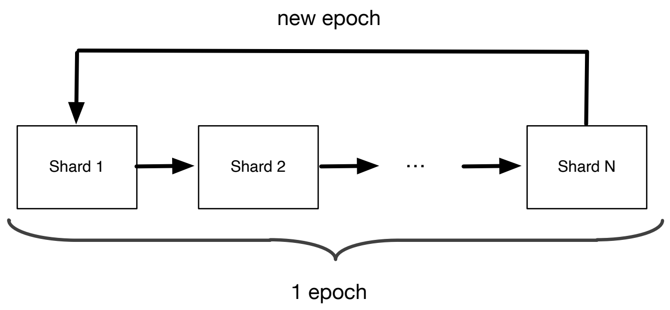

Fig. 3: Incremental training illustration.

cannot exceed the size limit of each blockchain block. The maximum size of a block, or the block gas limit, is 30 million gas for Ethereum. While the model complexity increases, the number of data samples allowed in each block will decrease.

Most ML algorithms are naturally incremental. That is, the data samples can be fed into the model in batches for training. Therefore, such incremental training will not impact the accuracy of the generated model. However, there are ML algorithms that cannot be trained incrementally without additional overhead or complexity. Examples of such models are k-NN and decision trees. These ML models often require to scan the whole dataset for each round that updates the model.

On-chain inference. On-chain inference has the advantage of ensuring that an inference operation is performed on the desired ML model that is maintained on-chain. The model that is maintained on-chain could be one that was trained on-chain (as discussed above) or off-chain (as we discuss in the next section).

On-chain prediction is straightforward as it entails processing an input sample using the trained model on-chain. This can be done by making a call to a function on-chain that implements the inference logic. This would lead to monetary fees that are proportional to the complexity of the inference process and the size of the data sample used as input. Also, this leads to overhead in terms of latency to wait for the smart contract call to be committed in a block.

Alternatively, a client can utilize a model that is maintained on-chain without the corresponding cost and latency overhead. This is because the inference process does not lead to changing the state of the smart contract. Therefore, the client can read the model information from blockchain and process the inference locally. The client, however, must ensure that the read model is the correct one by running as a blockchain client and observing the state of the smart contract directly. Running a full blockchain client can incur high overhead—due to needing to download the whole chain—which would prevent this approach. An alternative is to run a light blockchain client that reads from a full client.

C. Off-chain ML Training and Inference

Off-chain training and inference can significantly reduce the monetary fees and latency overhead of on-chain training and inference. Off-chain nodes are more powerful with strong computation capacity compared to smart contracts that are limited in size and compute capacity. However, off-chain nodes are not trusted, which undermines the original intention of using blockchain. Therefore, the main challenge in building off-chain training and inference solutions is to guarantee the integrity of ML models and predictions made by off-chain resources.

While off-chain nodes may take the responsibility of training and inference, the on-chain smart contract is still involved in various ways depending on the off-chain solution. Generally, the on-chain node's role in off-chain solutions is one or more of the following: (1) coordinate between off-chain nodes that are working collaboratively, (2) verify off-chain computation outcomes, (3) manage incentives and penalties of off-chain nodes, (4) maintain meta-information about the ML model and corresponding digests to prove their integrity. In the rest of this section, we provide details about these roles and how they correspond to off-chain solutions.

The right part of Figure 2 shows the taxonomy of off-chain approaches for ML training and inference. Off-chain ML training and inference can be categorized into single party-based and multiple party-based approaches. Single party-based solutions are ones that can generate a proof of the integrity of the outcome. This can be based on trusted hardware or software-level verification techniques. Multiple party-based solutions are ones that rely on performing the computation on a number of nodes to ensure that a malicious subset does not threaten the integrity of the computation.

In the rest of this section, we describe and discuss the advantages and limitations of single party and multi party approaches.

1) Single party-based methods: Single party approaches are ones that can rely on a single off-chain node to perform the computation (training and/or inference.)[1] The main challenge solved by these approaches is the ability to provide a proof of the outcome of the computation that is performed by the off-chain node. There are mainly two kinds of single party-based approaches: enclave-based and verifiable computation-based methods. While enclave-based methods leverage trusted hardware technology, verifiable computation-based methods concentrate on software-level verification.

Enclave-based method [42]. Enclave-based computation relies on Trusted Execution Enviroments (TEEs) to run programs while preserving the integrity of the computation. This is done using hardware-level isolation and memory encryption. Well-known secure enclaves include Intel's Software Guard Extensions (SGX) and AMD's SEV technology. Secure enclaves are also available on a number of cloud computing platforms such as Amazon AWS (AWS Nitro Enclaves) and Microsoft Azure.

Figure 4 shows the basic framework of the enclave-based method. Enclave-based methods follow a three-step approach.

[1]Here, we refer to being able to rely on a single off-chain for safety and correctness of operations. For liveness, these methods can be augmented with replication solutions.

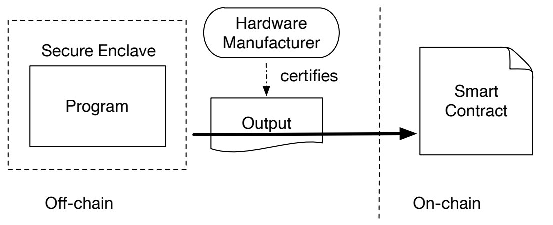

Fig. 4: Enclave-based method illustration.

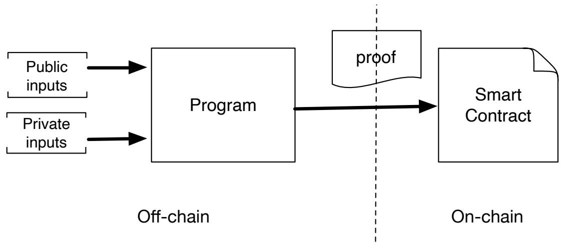

Fig. 5: VC-based approach illustration.

First, the off-chain node initializes a secure enclave. Second, the program to be processed in the secure enclave is loaded. Third, the off-chain node can start processing requests using the program in the secure enclave. The outcome of each computation running on the enclave has a proof (attestation) of correctness. This attestation can be used to ensure that the computation was performed correctly using a given program and input. Attestations can then be sent to the on-chain smart contract that maintains the meta-information and proofs for future clients.

Verifiable computation-based method (VC-based for short). The core idea of the VC-based method is to enable a client to verify that an untrusted off-chain node has performed computations correctly. The VC-based method involves using cryptographic techniques to enable the verifier, who may not have access to the data or the computation itself, to verify the correctness of the computation without actually performing the computation. Specifically, the off-chain node generates a proof that can be used to verify or attest the correctness of that computation (Figure 5).

There are several approaches to verifiable computing, including interactive proof systems, probabilistic proof systems, and fully homomorphic encryption [43]. These approaches use different methods to enable the verifier to check the computation, such as by allowing the verifier to ask questions about the computation or by encrypting the computation in a way that allows it to be evaluated without revealing the underlying data.

A popular VC-based method is based on Zero Knowledge Proofs (ZKP). ZKP allows a *prover* to produce a proof π that proves to a *verifier* that the result of a public function f on a public input x and secret input w of the prover is $y = f(x, w)$. ZKP guarantees that the *verifier* rejects incorrect proofs with overwhelming probability. An interesting feature of ZK proofs is that they reveal no extra information about the secret w beyond the result. ZKPs can be realized in the form of Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) [44], Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) [45], and Bulletproofs [46]. The generated proof with ZKP is typically designed to have a small size and can be efficiently verified. This makes ZKP methods suitable as an off-chain approach, since the on-chain verification can be a lightweight task.

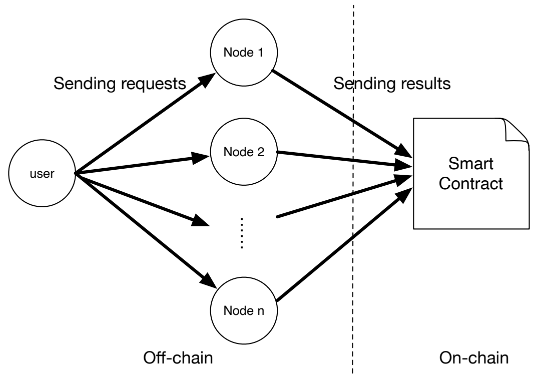

Fig. 6: Voting-based method illustration.

An assumption about using VC-based methods is that the program (which corresponds to the function $f(\cdot)$ mentioned above) needs to be public for ML on Chain tasks; the program is known to both the *prover* and *verifier*. This is typically the case for ML on Chain since the program corresponds to a public ML algorithm. The exceptions are cases when the used ML algorithm is proprietary.

Verifiable computing can be used in a variety of applications, including secure cloud computing, secure multi-party computation, and secure outsourcing of computations. It can also be used to ensure the integrity of computations performed in areas such as financial analysis, scientific research, and machine learning.

2) Multiple party-based approaches: The idea of multiple party-based approaches is to leverage multiple nodes to prevent a subset of potentially malicious nodes (referred to in the literature as byzantine nodes [34]) from threatening the integrity and correctness of results. The main assumption held by this group of solutions is that the number of malicious nodes (or failures) is bounded. This is most typically defined as assuming that no more than f byzantine nodes—out of a total n nodes where $n > f$—are part of the system. Another assumption in this approach is that off-chain nodes cannot fake new identities. This is important to prevent sybil attacks where a byzantine node acts with many fake identities. Typical multiple party-based approaches include voting-based methods [22] and incentive-based methods [13].

Voting-based method. In this group of methods, a number of off-chain nodes are asked to train a model with the same dataset and configuration. Then, the trained models from the off-chain nodes are collected. If there are enough received identical models, then the model is approved. This method works if the number of responses r for a given model is larger than f. In the following, we briefly describe a baseline voting-based method.

Assume that there are $2f+1$ nodes where f is the maximum number of byzantine nodes. The baseline solution to deal with byzantine nodes is to obtain the answers from multiple nodes. A collection of $2f+1$ nodes can submit their models to the smart contract. When the smart contract receives $f+1$ identical responses with model M, the smart contract returns M as the result. Since there are at most f malicious nodes, the returned M must be the outcome of correct computation.

The reason we need $2f+1$ off-chain nodes instead of only $f+1$ off-chain nodes is that byzantine nodes may act maliciously by being unresponsive. In such a case, if there is less than $2f+1$ nodes, and f nodes out of them are unresponsive they can block the whole system. Note that $2f+1$ is also less than the typical number of nodes used in byzantine agreement, which is $3f+1$ [34]. This is because the smart contract is a trusted party that disseminates information and collects votes. Therefore, unlike a traditional byzantine agreement protocol, nodes in a voting-based off-chain method are not trying to reach agreement on values—rather, they are only providing the outcome of a predefined computation that is disseminated and collected by a trusted party.

Figure 6 illustrates the baseline solution discussed above. A user sends requests to the nodes in order to generate a ML model. These n nodes will train the models independently and send their models to the smart contract on blockchain. The user needs to specify the ML workflow and the relevant parameters with which a node can generate the ML model. The supplied information must be enough for the ML training process to be deterministic. This deterministic nature of computation by off-chain nodes is important since any variations/randomness would lead to different final outcomes.

A challenge in the voting-based method is that an off-chain node may observe the response of another node and copy it—therefore, sending a response without having to make the computation. Off-chain nodes may be incentivized to do this as they would get the monetary reward of participating in ML training while not having to put resources for the actual computation. To overcome this problem, the nodes are required to send the signatures of their results to the smart contract to indicate that they already got the results. The nodes send the their outcome after the signatures of every node are received by the smart contract. In this way, an off-chain node cannot steal another node's response as the response will not match with the signature it sent previously.

Incentive-based methods. Incentive-based methods borrow concepts from game theory and mechanism design. The key idea of incentive-based methods is to encourage the off-chain participants to publish correct results to maximize their own profits. In such a setting, multiple off-chain nodes compete to submit their answers (trained ML model) to the smart contract. The first to submit their answer wins a reward. To check whether the submitted answer is correct, other off-chain nodes can challenge it. In the case that a submitted answer is incorrect, a challenger would get the reward of the answer instead of the first node that submitted the wrong answer. For this approach to work, there is a need for a *challenge period* after the answer is submitted to allow off-chain nodes to challenge the correctness of the answer. Only after an answer has been unchallenged for this challenge period, then it would be considered safe.

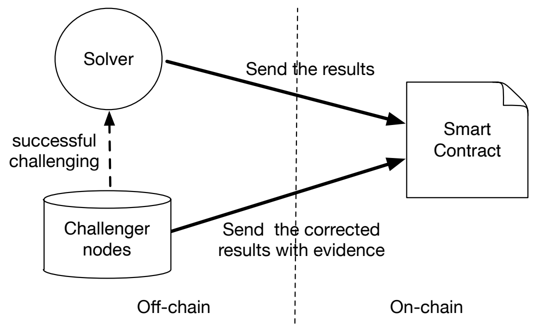

Fig. 7: Incentive-based method illustration.

Figure 7 shows the framework of the incentive-based method. A solver first sends the results of a computation task to the smart contract. Then, a challenger node sends the corrected results with evidence if a wrong result is detected [13]. Then, the smart contract judges whether the challenge is valid. If yes, the challenger node will then play the role of a solver. This process continues until no challenger nodes challenge the current results.

An example of this approach is in TrueBit [13], where the task is to compute the product of two matrices A and B. The *Solver* submits the answer C claiming that $A \times B = C$. The *Challenger* can challenge an incorrect answer by pointing to an error in one of the entries of C. Specifically, the Challenger shows that $c_{i,j}$ is not computed correctly, i.e., $c_{i,j} \neq \sum_{k=1}^{m} a_{i,k} \cdot b_{k,j}$. The challenger provides corresponding evidence that includes partial sums $d_0, d_1, \cdots, d_n$ where $d_n = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$. The smart contract verifies that i, j are coordinates, $d_n \neq c_{i,j}$ and $d_0 = 0$. If this does not hold, the *challenger* loses. Next, the *solver can defend their answer by providing k such that $d_k \neq d_{k-1} + a_{i,k} \cdot b_{k,j}$*. In this way, the smart contract only verifies the relationship between two consecutive partial sums are correct or not instead of calculating $\sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$. The *Solver* wins the game if the smart contract can verify this claim; otherwise the *Challenger* wins.

Another well-known example using the incentive-based method is the rollup [47], a layer-2 solution that aims to increase the scalability of the underlying blockchain by reducing the amount of data that needs to be processed and stored on-chain. Typical rollups include optimistic rollups, zk (zero knowledge) Rollups, and validium Rollups.

To perform a rollup, the smart contract first receives a batch of transactions from participating users. These transactions are typically grouped together based on some criteria, such as the type of asset being transferred or the identity of the sender. Next, the smart contract processes the transactions by executing the relevant logic and updating the state of the blockchain. For example, if the transactions involve the transfer of a digital asset, the smart contract would update the balance of the relevant accounts to reflect the transfers. Once the transactions have been processed, the smart contract creates a new transaction that summarizes the results of the rollup. This summary transaction, also known as a proof, is recorded on the main blockchain, along with any relevant metadata such as the block height and the root hash of the rollup.

In a rollup, a challenge is a mechanism that allows a party to dispute the validity of a bundled transaction or the proof that summarizes the results of the rollup. Challenges can be used to detect fraud or errors in the rollup, and to ensure that the rollup is being executed correctly. There are several ways that challenges can be implemented in a rollup, depending on the type of rollup being used. One common approach is to use an exit game, in which a challenger can prove that a bundled transaction is invalid by providing a valid proof that demonstrates the invalidity of the transaction. If the challenger is successful, they may be entitled to a reward, and the invalid transaction may be reversed or removed from the rollup. Another approach is to use a dispute resolution process, in which a neutral third party or a panel of experts reviews the challenged transaction and makes a determination on its validity. This process can be used to resolve disputes in a more formal and transparent manner. Regardless of the specific approach used, challenges are an important part of ensuring the integrity and security of a rollup, and can help to prevent fraud or errors from going undetected.

Distributed Byzantine ML (DbML for short). Distributed Byzantine machine learning refers to ML algorithms that are implemented in a distributed computing environment and are resilient to Byzantine failures. In a distributed computing system, there may be multiple nodes, or computers, that are working together to perform a computation. However, due to various reasons, some of these nodes may fail or behave maliciously, a scenario known as a Byzantine failure.

Distributed byzantine ML [48], [49], [50] was originally proposed to scale to large dataset and shorten the training time in byzantine distributed systems. Figure 8 illustrates a simple workflow of distributed byzantine ML. In distributed byzantine ML, the whole dataset is divided into N data samples where N is the number of nodes or machines. Distributed implementations of byzantine Stochastic Gradient Descent (SGD) typically takes the following form: a single trusted parameter server is in charge of updating the parameter vector, while the—potentially malicious—nodes perform the actual update estimation, based on the local data it possesses. More specifically, the parameter server executes learning rounds, during each of which, the parameter vector is broadcast to the nodes. In turn, each node computes an estimate of the update to apply (an estimate of the gradient), and the parameter server aggregates their results to finally update the parameter vector.

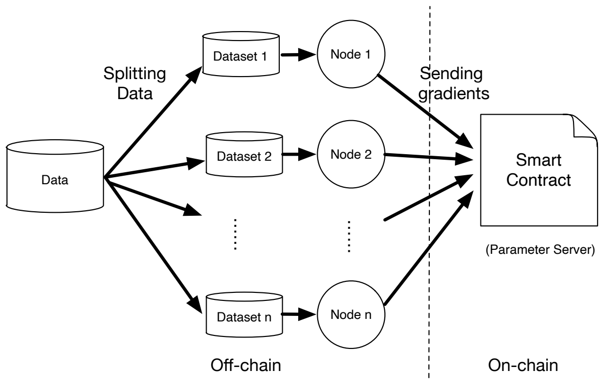

Fig. 8: Distributed byzantine ML illustration.

DbML assumes that there are byzantine nodes which may send malicious gradients to the parameter sever. Byzantine failures can occur when some of the machines in the system provide incorrect or malicious data to the learning algorithm. This can lead to incorrect or biased model outputs, which can have serious consequences in real-world applications.

The core idea of the distributed byzantine ML is to filter out malicious or incorrect data (or gradients) from potentially byzantine nodes, while prohibiting byzantine nodes from flooding the parameter server. Successful filtering of malicious input enables arriving to a correct model—with assumptions on the number of byzantine nodes f. Specifically, there are two kinds of approaches to addressing Byzantine failures in DbML. One approach is to use robust aggregation techniques that allow the system to identify and exclude faulty or malicious nodes from the computation. Another approach is to use techniques such as consensus protocols or verifiable computing to ensure the integrity of the computation.

DbML can be used in a variety of applications, including distributed training of ML models and distributed evaluation of ML algorithms. It is particularly useful in scenarios where the availability and reliability of the nodes in the distributed system cannot be guaranteed, such as in large-scale distributed systems or in systems with untrusted nodes.

In ML on Chain, distributed byzantine ML can be used by considering the smart contract as the parameter server and the off-chain nodes as the worker nodes computing gradients.

IV. Off-chain Methods Analysis

In this section, we present an analysis and discussion of the five off-chain approaches we described in the previous section. We consider the following aspects: trust assumptions, result integrity, and privacy. Table I summarizes the discussions. In addition to the analysis provided in this section, we present more discussions that are related to practical and performance-related aspects in Sections V (evaluation section) and VII (conclusion and future directions section).

TABLE I: Comparison between off-chain solutions for ML training and inference.

	Trust Assumption	Result Integrity	Data Privacy
Enclave-based	A trusted manufacturer	Guaranteed	Yes
VC-based	No trusted assumption or with at least 1 honest node for trusted setup	Guaranteed	Yes
Incentive-based	Nodes are rational to maximize their individual profits and at least a single challenger is available	Partially guaranteed	No
Voting-based	The maximum number of byzantine nodes f is known	Guaranteed	No
DbML	The maximum number of byzantine nodes f is known	Partially guaranteed	Yes

A. Trust Assumptions

VC-based methods realized in the form of zk-SNARKs typically need a trusted setup. The trusted setup refers to generating an explicit representation of the relation so that the verifier knows exactly what is being proven. This requires a trusted third party for the trusted setup. Multiparty computation (MPC) is often used in order to reduce the trust assumptions. Ideally, such a MPC involves a large number of nodes, and ensures that if at least one node behaves honestly, then the result is secure. After the trusted setup, the VC-based methods do not need trusted nodes any more for later computations. Some variants of the VC-based method does not need a trusted setup; however, they often suffer from high overheads of proof generation or verification [46].

The trust assumption of enclave-based method is that the manufacturer which provides the service is trusted [42]. This kind of trust is two-folds. First, we trust that the enclaves provided by the hardware manufacturer are safe so that program can not be tampered with when running in the enclaves. Second, we trust that the hardware manufacturer can honestly certify that the results originate from a secure enclave running the correct off-chain program.

DbML can tolerate f byzantine nodes with a set of n nodes in total where f is known. The state-of-the art DbML method [48] ensures resilience against a system with at most one third of the nodes being byzantine. But, when there are more than one third byzantine nodes, it can still achieve the convergence guarantee in term of ergodic convergence.

The voting-based methods can tolerate at most f byzantine nodes, when there are $2f + 1$ nodes in total. The incentive-based method assumes that the off-chain nodes (participants) are rational in the sense that they act to maximize their individual profits. If nodes want to maximize their profits, it means that a node will not be incentivize to lie and that challengers are incentivized to challenge. Particularly, the off-chain nodes expect fair compensation for their work on the computation task. Another implicit assumption of the incentive-based method is that at least a single challenger is available.

B. Results Integrity

The enclave-based method to guarantee the integrity of the results as long as the enclave are secure and the manufacturer is honest. Theoretically, the VC-based method cannot 100% guarantee the integrity of the results. However, it guarantees the integrity of the results with overwhelming probability (extremely close to 100%). The incentive-based method guarantees the integrity of the results as long as the challengers can point out any errors in the results. However, this does not always hold even the challengers are incentivized to challenge in practice. Therefore, the result integrity of the incentive-based method is partially guaranteed. The voting-based methods can achieve integrity of the results if there are no more than f byzantine nodes which is explained in Section III-C2. The DbML can only guarantee partial integrity of the results. Specifically, the results generated by DbML are only based on the datasets held by the honest nodes since the results submitted by the byzantine nodes will be filtered.

C. Privacy

From the view of the user requesting the ML task (called req-user), none of the five methods can guarantee the privacy of the datasets, ML models and predictions. Since the datasets, ML models and predictions are required to be sent to the smart contracts by the req-user, they are public to all blockchain users. There are some techniques [51] that can encrypt the inputs if the req-user want to protect the privacy of their datasets. These techniques are orthogonal to the ML on Chain approaches and can be applied to preserve information privacy. These techniques, however, may introduce some challenges in terms of performance overhead and/or impacting model accuracy negatively.

From the view of the off-chain node, all the five methods cannot guarantee the privacy of ML models and predictions as the models and predictions are eventually stored in smart contracts. However, the privacy of the datasets may be preserved in enclave-based, VC-based and DbML methods.

Enclave-based methods do not need to expose the inputs to others except for the manufacturer. As long as we trust the manufacturer, the privacy of inputs is guaranteed. Enclaves are used to protect the confidentiality of sensitive data by executing computations on the data within the enclave. This ensures that the data is not visible to anyone outside the enclave, and that it cannot be accessed or modified by unauthorized parties.

The VC-based method can protect the privacy of inputs of the ML tasks. This includes ZKB-based methods which can take both public inputs and private inputs for a request. ZKB provides a way for one party to prove that they know something without revealing what that something is, which protects the privacy of sensitive data in a variety of situations. However, directly setting the sensitive data as a private input may not lead to the correct result for ML on Chain tasks. For example, if the dataset is taken as the private input for ML training task, the ZKB-based method can prove that there exists a dataset (i.e. our private dataset in this case) such that the model is the output after training. Nevertheless, this proof does not indicate that the model is the output obtained when running it on the specific dataset (i.e. our private dataset). To this end, for ML on Chain tasks, the off-chain node needs to prove that the input data is consistent, e.g., with some public commitment that others trust is a commitment to the off-chain node's dataset. Therefore, both the dataset consistency proof and the integrity of models (or predictions) should be proved if the off-chain node needs to protect the privacy of the private dataset.

In DbML, participating off-chain nodes train a model on their own dataset and then send the model updates back to a central server. This process helps protect data privacy because the data never leaves the control of participating off-chain nodes. The model updates are sent to the the smart contract, but the data itself is not shared. This means that the off-chain nodes can collaborate on ML tasks without exposing sensitive data to others. However, it is possible that the gradients of DbML could leak information about the data used to compute them because they are derived from the data and contain information about the patterns and relationships present in that data [52]. For example, if the model is able to accurately predict the development of the medical condition, it is likely that the gradients will contain information about the patterns and relationships in the data that were used to make those predictions. This could include information about the relationship between certain medical conditions and certain demographic characteristics, such as age or gender.

For the voting-based methods, the privacy of the inputs can not be guaranteed. This is because each node has to take the same inputs so that all the honest nodes can generate the same outcomes. Therefore, the inputs are public to all nodes. Similarly, the incentive-based methods need to keep data public so that challengers and solvers can performed the tasks related to challenging and defending answers.

Privacy is a crucial aspect for ML on Chain tasks. When the req-users cannot provide ML training algorithms or models, the off-chain nodes need to propose their own solutions. If the off-chain nodes do not want to make their solutions public to others (referring to taking the ML training algorithms and models as inputs), the two multiple party-based methods—voting-based and incentive-based methods—are not applicable for such tasks.

D. Storage

The datasets for training and inference take up a lot of storage for ML tasks. While storing these datasets on blockchain is impractical, existing applications often use a decentralized storage system with redundancy, e.g., IPFS [53], SWARM [54], Storj [55], and Sia [56], to improve the integrity and availability of the datasets.

One key advantage of decentralized storage systems is that they can be more resilient to failures or attacks, as the data is distributed across multiple nodes and can be accessed from any node on the network. This can make decentralized storage systems less vulnerable to data loss or downtime due to server failures or other issues. Decentralized storage systems can also offer greater privacy and security for users, as the data is not stored in a central location that can be easily accessed by third parties. In addition, decentralized storage systems may offer users more control over their data, as they can choose which nodes to store their data on and have more visibility into how their data is being used. These systems use various technologies and approaches to achieve decentralized storage, including distributed hash tables, blockchain technology, and peer-to-peer networking.

V. Experimental Evaluation

In this section, we perform an experimental evaluation of the different ML on Chain approaches we discussed.

A. Setup

Experimental setup. our experiments are performed on the Ethereum Goerli test network, which now has switched to proof-of-stake (PoS). We implement the on-chain components using solidity smart contracts, and implement off-chain components using python and C++. Software and libraries that we use for specific approaches are mentioned later in the section. The experimental environment is a computer with a Quad-Core Intel Core i5 processor, 8GB memory, running macOS Catalina. We use *ZoKrates* [12] to implement a zkSNARKs-based approach. The Groth16 [57] scheme is used to derive proofs with a small size.

ML models. We experiment with three ML models: **Logistic Regression**, **k-NN** (k-nearest neighbors) and **Neural Network**. **Logistic Regression** is a generalized linear model which is represented by the coefficients in linear combination. k**-NN** is a non-GD based model where the label of an object is based on the values of the k nearest neighbors. **Neural Network** is a non-linear model using interconnected group of neurons for computation.

k-NN model does not have a real training phase. This is because the prediction of each new data item is calculated based on all existing data items rather than a trained model. In other words, the inference of **k-NN** includes both training and inference phases. Neural Network is popular because of its high accuracy; however, training a **Neural Network** takes a lot of computation overhead. Our experiments with the above three models tend to show the concerns and challenges when we design decentralized applications with different ML models.

Datasets: To give us more flexibility to evaluate performance and monetary cost, we use a synthetic dataset in most experiments. Using a synthetic dataset allows us the flexibility arbitrarily vary the experiments conditions such as the number of features and data samples. We also perform an evaluation with two real datasets in Section V-F.

Compared approaches. We compare an on-chain ML baseline and five off-chain methods. Each off-chain method represents one of the groups we presented:

- **SGX-based:** We use the Software Guard Extensions (SGX) secure enclaves for the *enclave-based methods*.
- **zkSNARKs-based:** We select a zkSNARKs-based approach (with *ZoKrates* [12]) from the *VC-based methods* for experimenting as zkSNARKs have a small proof size and low verification cost.
- **Incentive-based:** We implement a representative incentive-based solution that mimics the three-phase evidence-challenge pattern of TrueBit [13]. We set the evidence length statically to be 100 characters. The specific length depends on the evidence-challenge strategy and used ML model.
- **Voting-based:** We use the baseline voting-based method described in Section III-C2. The smart contract receives the results with signatures from off-chain nodes.
- **Parameter server-based (ParameterS-based for short):** Instead of using a traditional server in *DbML* [48], we take the smart contract as the parameter server maintaining the ML model.

We set the number of off-chain nodes to 9 for multiple party-based methods (i.e. incentive-based, voting-based and ParameterS-based methods). We set the rounds of information exchange for the incentive-based method to 5.

Default parameters. Unless we mention otherwise, the number of features is set to 10 and the number of data samples for inference is set to 100. The number of epochs for training Logistic Regression and Neural Network is 10. The number of layers for Neural Network is 3.

Cost. In Ethereum, on-chain execution and verification cost is calculated in a unit called *gas*. For ease of exposition in the rest of this section, we also present the cost in dollars. Because the gas-dollar conversion rate fluctuates, we make the following assumption about the price of gas. We assume the base gas price as 20 Gwei[2] according to recent approximate pricing on Ethereum Mainnet at the time of writing this paper. We also assume that the price of one ether is equal 1500 dollars.

B. On-chain execution cost evaluation

We evaluate the cost of training and inference for on-chain solutions of Logistic Regression, k-NN and Neural Network models. Figure 9 illustrates the gas fees (dollars) for three ML models in terms of training. The cost of on-chain training increases almost linearly with the size of the training samples and the size of features. The k-NN model does not have a training phase; its cost is set to 0. The cost of Neural Network is much higher than that of Logistic Regression due

[2]Gwei is a denomination of Ethereum's ether (ETH). A gwei is one-billionth of one ETH.

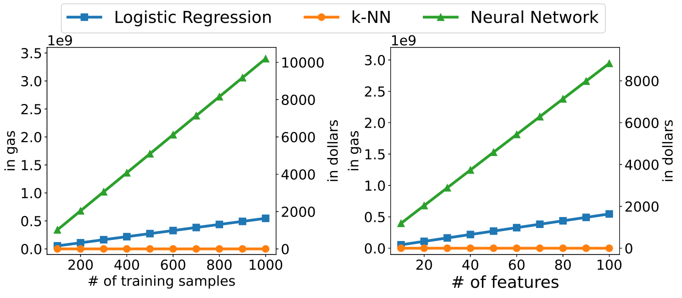

Fig. 9: On-chain execution cost for on-chain training.

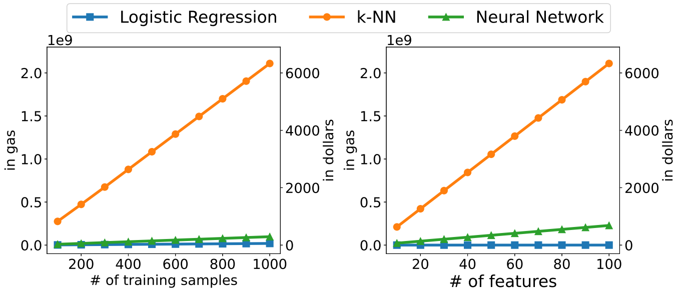

Fig. 10: On-chain execution cost for on-chain inference.

to its higher training complexity when the number of training samples or features increases. This indicates that the on-chain cost almost linearly increases with the algorithm's complexity for training.

Figure 10 shows the cost of the three ML models for inference. As the inference for the k-NN model is not preceded with a "training" phase, its cost is significantly higher than both Logistic Regression and Neural Network especially when the number of training samples and features become large. The cost for a single data item inference is relatively cheap for Logistic Regression and Neural Network models. However, the number of data items for inference in practice can be quite large, leading to quite high fees.

C. On-chain verification cost evaluation

Training. Although off-chain methods do computation with off-chain machines, they need to send the computation results to the smart contract for verification. We show the components of on-chain verification cost in Table II. Since k-NN does not have a training phase, we do not include it in the table.

As can be seen from the subfigures on the left of Figures 11 and 12, the on-chain verification cost is a constant with respect to the number of data samples for training tasks. This is because increasing the number of training data will not change the size of the model (i.e. the number of coefficients or parameters). As the models need to be stored in the smart contract, the minimal verification cost (which is quite close to that of the SGX-based method) for the five methods is for receiving (or storing) the models. The ParameterS-based method gets the highest cost for both Logistic Regression and Neural Network as it needs to store multiple rounds of gradi-

TABLE II: Main factors that affect the on-chain verification and off-chain running time for five off-chain methods.

	On-chain verification cost	**Off-chain running time**
SGX	Receiving and storing the results with a certificate from a single node	(1) Running time for the program in SGX-based enclaves (2) Time for creating the enclaves and data exchange between enclaves and outside environment
zk-SNARKs	Verifying the received proof from a single node	(1) Time for proof generation (2) Time for keys generation
Incentive	(1) Receiving and storing results from a single node (2) Receiving and storing evidence from a single node in multiple rounds	Sum of the running time for each round of challenge
Voting	(1) Receiving and storing results and signatures from multiple nodes (2) Similarity computation for models and predictions	Maximal running time of the program execution in multiple nodes
ParameterS	(1) Receiving and storing gradients and models from multiple nodes (2) Maintaining or updating the ML model	Sum of the maximal running time of the program execution in multiple nodes for each round of updating

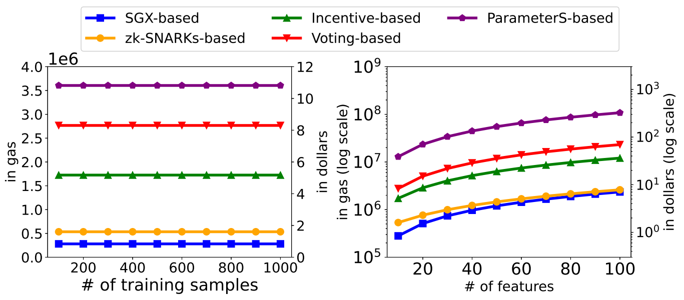

Fig. 11: On-chain verification cost for Logistic Regression training.

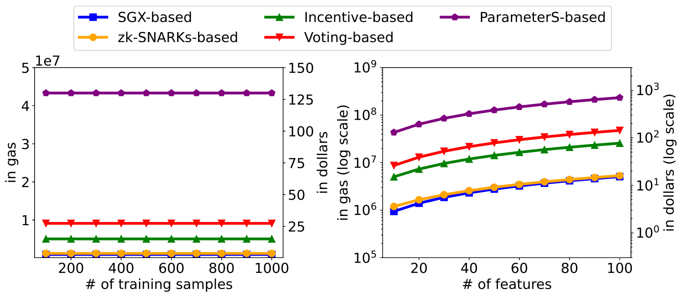

Fig. 12: On-chain verification cost for Neural Network training.

ents (left part of Figure 11). Because the large size of Neural Network models compared to Logistic Regression models, the cost of the parameter server-based method for Neural Network is much higher than that of Logistic Regression (Note that the size of gradients equals to that of Neural Network models).

The subfigures to the right of Figures 11 and 12 show that when the number of features increases, the cost of all the five methods grows. This is because the size of the model increases which means that more is stored/communicated to the smart contract. The incentive-based, voting-based and ParameterS-based methods have the highest verification cost because the smart contract needs to store the models received from multiple nodes. Since the ParameterS-based method needs to store the gradients from multiple nodes in multiple rounds, and the number of gradients increases with the number of features, it has the highest cost. The SGX-based method only incurs the cost of storing one model therefore it has the lowest cost. The zk-SNARKs-based method also shows low verification cost and approaches the cost of the SGX-based method when the number of features is large. Compared with the cost with Logistic Regression models, Neural Network training incurs more cost because the model has a larger size.

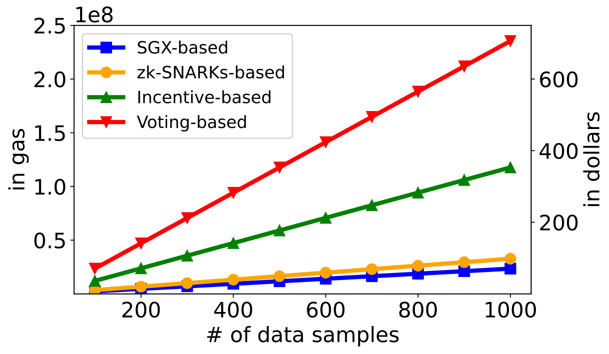

Fig. 13: On-chain verification cost for three ML models inference. (The cost of the three models is the same.)

Inference. We report the verification cost for four methods without including ParameterS-based. This is because ParameterS-based handles the training process only, and the resulting ML model can then be utilized for inference in any of the four other off-chain approaches. For all methods, the verification cost does not change when varying the number of features. This is because the size of predictions does not change with a different number of features. Table III shows the on-chain verification cost for inference task with three ML models. While the incentive and voting-based methods need to store the multiple predictions, they take relatively high cost compared to the SGX and zk-SNARKs-based methods. The

TABLE III: On-chain verification cost for inference (in dollars) with three ML models (for 100 data samples).

	Logistic Regression	k-NN	Neural Network
SGX	7.045	7.045	7.045
zk-SNARKs	10.465	10.465	10.465
Incentive	35.225	35.225	35.225
Voting	63.405	63.405	63.405

TABLE IV: Off-chain running time (in seconds) for Logistic Regession training with different amounts of training samples

	100	500	1000	5000	10000
SGX	0.059	0.066	0.070	0.074	0.079
zk-SNARKs	0.530	14.611	41.869	1315.43	6754.27
Incentive	0.031	0.039	0.064	0.709	0.109
Voting	0.006	0.007	0.012	0.014	0.021
ParameterS	0.004	0.005	0.006	0.008	0.011

TABLE V: Off-chain running time (in seconds) for Neural Network training with different amounts of training samples

	100	500	1000	5000	10000
SGX	0.057	0.081	0.096	0.185	0.269
zk-SNARKs	6.484	75.522	264.30	6523.21	26378.93
Incentive	0.083	0.156	0.183	0.556	1.143
Voting	0.021	0.042	0.055	0.131	0.219
ParameterS	0.011	0.025	0.034	0.054	0.115

TABLE VI: Off-chain running time (in seconds) for k-NN inference with different amounts of training samples

	100	500	1000	5000	10000
SGX	0.061	0.063	0.063	0.068	0.076
zk-SNARKs	0.215	0.537	1.968	60.35	268.38
Incentive	0.004	0.011	0.036	0.109	0.189
Voting	0.001	0.002	0.006	0.022	0.038

cost of zk-SNARKs-based method take around 50% more than SGX-based method as its verification cost is also linear with the size of the input (the size of its input is 100 in the table). The cost of SGX is mainly due to storing the predictions (results).

Figure 13 shows that the verification cost increases for all the four methods under the three ML models when the number of data samples increase. This is because the size of predictions increase almost linearly with the number of data samples. The cost for all the three ML models is the same. The SGX and zk-SNARKs-based methods achieve a low cost compared to the other methods as the smart contracts in these two methods only need to store predictions from a single node once.

D. Off-chain running time evaluation

In this section, we measure the off-chain running time of the solutions we compare. Because zk-SNARKs-based methods take significantly high running time when the model is complex, we set the number of epochs to 1 for Logistic Regression and Neural Network training for evaluating the five off-chain methods. In this way, we can show the characteristics of running time for zk-SNARKs when the computation task is light.

Training. We summarize the key factors that impact the off-chain running time in Table II and show the results in Tables IV to VII. Table IV shows the results for Logistic Regression training; in multi-node solutions, we report the maximal running time in all the nodes. The voting-based method has the least running time since its computation is often completed in parallel with multiple nodes. The incentive-based method need to sequentially propose challenges, therefore its running time is nearly n times of that of the voting-based method where n is the number of nodes. For the SGX-based method, we load the whole dataset into the enclave only once (we discuss this more later in the paper). The SGX-based method takes a little more time than the voting-based method as it needs to create an enclave. The zk-SNARKs-based method has a relatively small running time when the dataset is small; however, its running time increases significantly with the number of training samples.

A trend similar to the above results with Logistic Regression is also observed for Neural Network training as illustrated in Table V. Because the ParameterS-based method splits the dataset for processing and computing, it has the least running time. The zk-SNARKs-based method takes more time for Neural Network than Logistic Regression with the same dataset as Neural Network has higher complexity. Table VI shows the off-chain running time for k-NN inference when predicting a single data sample. Because k-NN performs the virtual training phase when predicting a data sample, the results shown in Table VI can be taken as the training time for the four off-chain methods. Compared with Logistic Regression and Neural Network training, k-NN takes less time for training for all the four off-chain methods.

Inference. As the ParameterS-based method is not applicable for inference, we show the running time for the other methods in Table VI to VII. The running time with the zk-SNARKs-based method is always much longer than others. While the off-chain running time for Logistic Regression inference seems close to that of k-NN by comparison between Table VI and Table VIII, it does not mean that k-NN inference is as cheap as that of Logistic Regression. Table VI illustrates the running time for predicting a single prediction sample (note that we vary the number of training samples rather than the number of prediction samples in Table VI), while Table VIII shows the running time for different amounts of prediction samples. Table VII indicates that the Neural Network inference takes more time than that of the Logistic Regression inference

TABLE VII: Off-chain running time (in seconds) for Logistic Regression inference with different amounts of prediction samples

	100	500	1000	5000	10000
SGX	0.059	0.062	0.065	0.072	0.079
zk-SNARKs	0.163	0.498	2.836	58.45	427.73
Incentive	0.004	0.021	0.041	0.118	0.206
Voting	0.001	0.004	0.008	0.023	0.042

TABLE VIII: Off-chain running time (in seconds) for Neural Network inference with different amounts of prediction samples

	100	500	1000	5000	10000
SGX	0.064	0.073	0.081	0.134	0.189
zk-SNARKs	0.327	0.752	108.95	1396.21	6076.84
Incentive	0.024	0.092	0.159	1.408	0.541
Voting	0.005	0.018	0.032	0.083	0.128

because the more complexity of Neural Network models. All in all, the cost of ML inference becomes considerable when the prediction samples are large, even though the cost of ML inference for a single prediction sample is little relatively. Besides, the zk-SNARKs-based method always takes much more time than the other three methods.

E. Scalability and trade-off evaluation

We now evaluate the scalability and trade-offs for the five off-chain methods.

SGX-based: When the dataset becomes quite large, ML models are usually trained in batches. In this case, data is frequently loaded to the enclave. One round of data exchange represents that the data is loaded to the enclaves once. Figure 14 shows that the time cost almost linearly increases with the batch size (i.e. the number of data samples) and the number of rounds of data exchanges. Either 100 thousands data exchanges with 10 thousands data samples or 1000 data exchanges with 1 million data samples take no more than 6s.

zk-SNARKs-based: The ML training algorithm often includes multiple epochs of iterations. As can be seen from Figure 15(a), the running time for generating proofs increases significantly when the number of epochs become larger. One way to overcome this overhead is to split the training tasks into several subtasks. For example, each epoch of computation is taken as a subtask and we denote the proof size as one unit for an epoch of computation. In this way, as shown in Figure 15(b), the running time for the whole task can be significantly reduced while the on-chain verification cost grows in a slower pace.

Voting-based: The on-chain verification cost of the Voting-based method includes two parts. One is the cost for receiving

TABLE IX: Cost evaluation for the PrameterS-based method on Neural Network training (with $40K$ training samples) with differet numbers of off-chain nodes.

# of off-chain nodes	**4**	**8**	**12**	**16**
On-chain verification cost) (in dollars	30.096	60.192	78.288	120.384
Off-chain running time (in seconds)	4.883	2.452	1.626	1.218

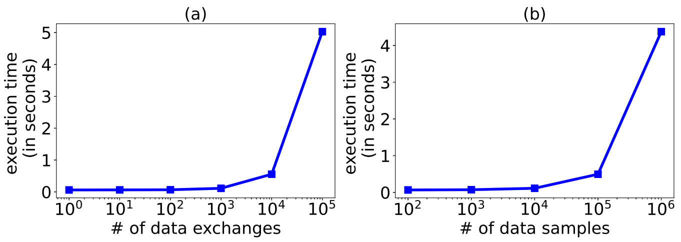

Fig. 14: The scalability of SGX-based method (the default number of data samples is 10000 for figure (a) and data exchanges is 1000 for figure (b).

models or predictions from multiple nodes (9 in our experiments), the other one is for computing the similarity between models or predictions. Figure 16 shows that the cost of storing results (i.e. models and predictions) for training and inference tasks is always higher than that of similarity computation. And the cost of model/prediction similarity computation becomes negligible compared with the model/prediction storage cost when the number of features/samples becomes large.

The **ParameterS-based** method has the lowest off-chain running time; however, increasing the number of off-chain nodes will also increase the on-chain verification cost nearly linearly. This observation can be seen from Table IX. In addition, when the number of off-chain nodes increases, the off-chain running time reduces as each off-chain node trains on a smaller dataset (assuming that the whole dataset is split uniformly into small dataset and each off-chain node will be assigned with a small dataset).

For the **Incentive-based** method, both the on-chain verification cost and off-chain running time rely on the number of challenges that appear. If every challenge takes the same on-chain cost, then the overall on-chain cost is linear with the number of challenges. However, it is difficult to predict what a challenge will look like for a specific ML task.

F. Evaluation with real datasets.

We evaluate the impact of real datasets in terms of on-chain execution cost and on-chain verification cost. Due to the space limit, we only show the results of training a Neural Network. The first real-world dataset, Amazon Access Samples (Access for short) [58], is an anonymized sample of access provisioned within the company. It contains more than $710K$ data samples. We use two numeric features in the

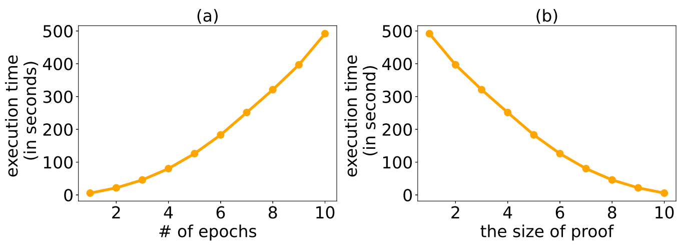

Fig. 15: The trade-off of the zk-SNARKs-based with Neural Network.

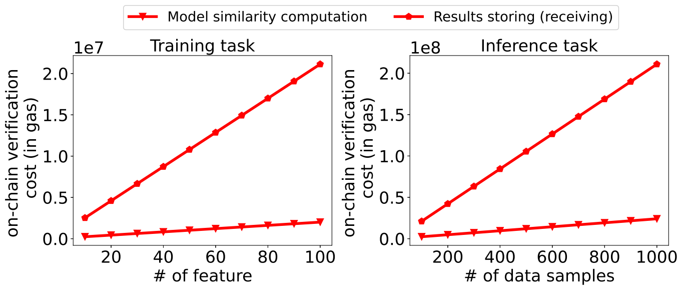

Fig. 16: The comparison between model similarity computation and model storing for the voting-based method with Neural Network.

dataset for the evaluation.The second real-world dataset, 3D Road Network (Road for short) [59], includes 3D road network with highly accurate elevation information. It contains $430K$ data samples. For a fair comparison, we create two synthetic datasets (all the values are set as 1) with the same size (both the number of features and training samples) of the two real datasets.

As can be seen from Table X, the difference between the average on-chain execution cost is negligible (less than 0.1%). The two real datasets have large values compared with those in the synthetic dataset, therefore leading to a little higher cost. The on-chain verification cost is similar with both real and synthetic datasets. This is because the parameters of models all fall in the range $(0, 1)$ no matter which values the datasets have.

VI. Related work

ML on Chain. To the best of our knowledge, there is no prior work that shares the goal of this paper—i.e., providing a taxonomy and understanding of the space and design tradeoffs of performing machine learning in blockchain applications. Existing work that is most relevant to ML on Chain tasks utilize ML for solving blockchain related problems without discussing the integrity and/or cost of executing ML programs [60], [37], [61], [7], [62], [8]. Others propose blockchain systems with relaxed security/trust guarantees to enable computation-intensive and storage-intensive tasks to be performed on chain [63], [64], [65], [66], [67]. While some works [39], [68] summarize general off-chain approaches, they

TABLE X: Evaluation with two real datasets in term of cost (in gas). The on-chain execution cost is about the average cost of a single data sample. The SGX-based method is used for on-chain verification cost evaluation.

# of features for training	Dataset	On-chain execution cost	On-chain verification cost
2	Access	16954731	382918
	Synthetic	16954452	382918
4	Road	23082410	666267
	Synthetic	23063714	666267

do not target ML on Chain and lack quantitative analysis with experiments.

The five off-chain approaches we considered in ML on Chain have been also used or proposed for problems other than blockchain-based applications. The voting-based and incentive-based methods are often used for solving truth finding problem in the context of web applications [69], [70] and crowdsourcing [71], [72]; in these works, they are combined with probabilistic models, monetary rewards and/or punishment. The enclave-based and VC-based methods are usually utilized when both confidentiality and integrity need to be guaranteed. The use cases include ML applications [73], databases [74], and cloud computing [75]. DbML is particularly used for ML training [41] when the datasets need to be confidential or the dataset is too large to train efficiently.

Off-chain scaling and computing solutions. Off-chain scaling solutions are methods for increasing the capacity of a blockchain beyond its current limits. Off-chain scaling solutions related to ML on Chain mainly include blockchain oracles and layer-2 solutions. Blockchain oracles [76], [77], [78] are trusted third parties that constitute the interface between blockchains and the real world. The techniques used for guaranteeing the integrity of the data from real world in blockchain oracles can be covered by the five kinds of off-chain solutions we considered.

Layer-2 solutions [79] are built on top of the main (or layer-1) blockchain. Typical layer-2 solutions consist of Plasma [80] sidechain [81] state channels [82] Rollups [83] and TrueBit [13]. While each of these is solving a different problem, these layer-2 solutions combine both off-chain state and off-chain computations in arbitrary ways. In other words, these layer-2 solutions are hybrid off-chain solutions which are designed for improving the scalability of a specific task.

Different from layer-2 solutions, off-chain computing [39], [68] operate in different ways and have different implications for the underlying blockchain. In this work, the discussed five off-chain methods are more related to off-chain computing. Specifically, off-chain refers to the use of external computing resources to perform certain tasks or process data that are not directly related to the blockchain. This can include tasks such as data storage, computation, and communication, which are performed off the blockchain but may still be related to transactions or data on the blockchain.

Surveys on discussing ML on blockchain. Some existing surveys [63], [37], [84], [8], [85] investigate the ML techniques and applications that are combined with blockchain. The key insight of these works is that ML and blockchain can benefit from each other. One one hand, these surveys demonstrate that ML can help improve DApp operation and functionality [86], [35], [36] and utilizing models to gain insights from collected data [37], [61], [7], [62] at a high level. On the other hand, these surveys broadly demonstrate that involving blockchain in the process of ML training and prediction enables DApps to utilize ML models to improve security, traceability, transparency and accountability.

However, none of them investigate or evaluate the the detailed solutions and trade-offs for implementing ML tasks on blockchain. Our work, instead, provides a comprehensive analysis of the ML on chain space and the various approaches that can be utilized.

VII. Conclusion and Future Directions

In this section, we discuss the five ML on Chain off-chain approaches in terms of their advantages, limitations and future directions.

Encalve-based: The Enclave-based methods are the most appropriate ones for computation-incentive tasks. They have the lowest on-chain verification cost and introduce relatively low overhead in terms of latency. The main issue of the enclave-based computation is that the client or consumer has to completely trust the manufacturer. Once the manufacturer becomes byzantine, the integrity of the results can not be guaranteed. In addition, the enclave-based scheme allows universal computations but has potential security issues [87].

There are two potential directions in which enclave-based methods could evolve in the future. One possibility is the development of more advanced hardware-based security features that can provide even stronger protection against attacks. For example, we could work on the hardware-based cryptographic acceleration, which aims to design specialized hardware components that can perform cryptographic operations much faster and more efficiently than software-based implementations. This could make it even more difficult for attackers to compromise the security of enclave-based systems.

Another potential direction for the development of enclave-based methods is the use of machine learning and artificial intelligence (AI) techniques to enhance security. For instance, ML algorithms could be used to analyze patterns of activity within an enclave and identify potential threats or anomalies [88]. AI-based intrusion detection systems could also be developed to monitor enclave-based systems and alert administrators to potential security breaches [89].

VC-based: The non-interactive VC-based methods work well when the task involves little computation. They have relatively low on-chain verification cost. However, the general-purpose VC-based methods take considerable time for generating the proofs when the program is computation-intensive. While some works proposed customized VC-based methods for dealing with specific tasks, such as decision trees [90] and Neural Network inference tasks [91]. Some other works [92], [93], [94] aim to distribute the proof generation across machines in a compute cluster. Although they still suffer from high overheads or do not work for heavy computation tasks such as ML training, our point of view is optimistic as the area of efficient proof systems sees tremendous progress in terms of real-world deployment [95].

One possibility to non-interactive VC-based methods in the future is the development of more efficient and scalable zk-SNARK constructions, which could make it more practical to use these methods in a wider range of applications. For example, it is possible to design customized non-interactive VC-based methods and make trade-offs between overheads of proof verification and proof generation for specific tasks so that the proof generation overhead can be significantly reduced.

Incentive-based: The incentive-based methods are widely used on blockchain (e.g. Rollups [83] and TrueBit [13]). They could have quite low on-chain verification cost when the nodes are rational to maximize their individual profits. Although some incentive-based approaches are proposed for some specific tasks (e.g. matrix multiplication), enabling the smart contract to judge the correctness of complicated computation tasks, such as ML training, still needs to guarantee both the theoretical soundness and low overheads of verification.

There are two potential directions that the incentive-based methods for ML on Chain may take in the future. First, developers may work to improve the scalability of these methods so that they can be used more widely as the incentive-based methods can be resource-intensive for proposing and verifying challenges, which can limit their scalability [96]. Second, there is likely room for innovation of creating different models for how incentives can be structured for ML on Chain. Such new models and structures, therefore, enable reducing the size and computation overhead of evidence and challenges efficiently.

Voting-based: The voting-based method have low off-chain running time as the computation can be done in parallel. There are three main limitations of the voting-based methods: (1) the integrity of the results can not be guaranteed when the number of byzantine nodes exceeds the predefined value. Although increasing the number of nodes help mitigate this problem, it also incurs extra overheads due to additional redundancy. (2) the honest nodes may not deliver a deterministic result for a specific ML task, leading to the failure of receiving enough identical results (models). This can happen when there are some randomness or non-specified parameters in the ML algorithms. (3) the on-chain verification cost is quite high due to storing multiple results.

As a future direction, we may aggregate the models/predictions on an off-chain node before sending it to the smart contract. In this way, the on-chain verification cost can be significantly reduced. Also, the future voting-based methods should be able to deal with randomness in ML so that they can accept slightly different $f+1$ models. One technology in that direction is the ability to judge if two models are similar and trained on the same dataset even if the models are not identical. In addition, we may use more advanced voting systems that incorporate elements such as reputation or stake-weighted voting to ensure that the voting process is fair and representative for all the off-chain nodes.

DbML-based: Using DbML-based methods is a favorable approach when the datasets are owned by different users who are not willing to share their datasets. This is because the training process can be performed without coordinating between the different users.

Thre are a number of limitations for DbML. First, although DbML may guarantee convergence of the ML model, the model is only trained based on the datasets hosted by the honest nodes. Second, DbML-based methods are not applicable to non-GD based ML algorithms when the algorithms do not aim to optimize the loss functions. Some multiple-party computation (MPC) protocols are proposed to enable a set of nodes to jointly compute private inputs like DbML, they can not guarantee safety and liveness without strong assumptions on the number of corrupted nodes [97], [98]. Third, most existing methods assume that an honest majority can be guaranteed. However, the number of malicious nodes are often unknown in practice. Forth, the DbML-based methods have potential data privacy issues [99], [100] as data information may leak via the public gradients. Lastly, the on-chain verification cost of the DbML-based methods are quite high due to the cost of storing multiple gradients.

Future directions that could be pursued in the field of DbML for ML on chain include: (1) Developing ML algorithms that are robust to Byzantine failures: We could work on developing ML algorithms that are resistant to the influence of Byzantine nodes, either by explicitly designing the algorithms to be robust to such failures or by using techniques like fault-tolerant consensus protocols to mitigate the impact of Byzantine nodes. (2) Investigating the use of cryptographic techniques in DbML: Cryptographic techniques such as secure multi-party computation and homomorphic encryption [101] could potentially be used to enable DbML in more secure and private settings. In this way, off-chain nodes can aggregate the gradients and only send the final models/predictions to the smart contract, significantly reducing the on-chain verification cost.

ML on Chain: Adopting ML on blockchain has the potential to enable a wide range of decentralized applications that are more intelligent and efficient than those that rely solely on blockchain technology.

The benefits for ML on Chain include: (1) Improved decision-making and optimization: By integrating machine learning with blockchain, it may be possible to build more intelligent and efficient decentralized systems that are able to make better decisions and optimize their operations. (2) Increased security and privacy: Machine learning on blockchain could potentially be used to improve the security and privacy of decentralized systems by enabling the use of secure multi-party computation and homomorphic encryption techniques. (3) Greater scalability: ML on blockchain could potentially enable decentralized systems to scale more effectively by allowing them to process large amounts of data and make decisions more efficiently.

Neverthless, there are a few drawbacks for ML on Chain including: (1) High complexity: Implementing ML on blockchain can be a complex task, requiring expertise in both ML and blockchain technologies. (2) Limited data availability: The decentralized nature of blockchain systems can make it challenging to access and use data for ML purposes. (3) Potential for bias: As with any ML system, there is a risk that ML on blockchain could be biased if the data used to train the model is biased.

Future directions that could be pursued in the field of ML on chain include: (1) Succinct requests: The format of the task requests is non-trivial especially when the request takes large datasets as input; for example, a ML inference request with a large dataset. Directly placing the large size inputs to the blockchain will incur high gas cost. Therefore, an economic and effective way to represent the task request is crucial for ML on Chain tasks. Utilizing decentralized storage might be a good choice to overcome this overhead. (2) Succinct outputs: Similar to reasons of succinct requests, we may look for efficient ways to represent the outputs of ML models or predictions; for example, we may store a digest of the results or the compressed results on blockchain. (3) Developping efficient ML models: While performing a monolithic ML task might take overwhelming overhead, breaking up a monolithic ML computation into sub-tasks may significantly reduce the overhead of off-chain computing methods. For example, performing small ML computation with zk-SNARK can be highly accelerated with low overhead [102]. Therefore, we could work on developing efficient and scalable ML algorithms, such as incremental ML [103], [104] and online ML [105], [106]. (4) Hybrid Solutions: While each of the five off-chain methods discussed in this paper have their own advantages and drawbacks, there is likely room for innovation of creating a hybrid solution by combining some of these five off-chain computing methods.

Overall, the ML on Chain field is still in its early stages and there is a wide range of potential applications for this technology. It is clear that ML on blockchain has the potential to offer a wide range of benefits, but there are also challenges that must be addressed in order to realize its full potential.

References

[1] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *arXiv preprint arXiv:2101.08778*, 2021.

[2] Q. Yang, Y. Zhao, H. Huang, Z. Xiong, J. Kang, and Z. Zheng, "Fusing blockchain and ai with metaverse: A survey," *IEEE Open Journal of the Computer Society*, vol. 3, pp. 122–136, 2022.

[3] H.-j. Jeon, H.-c. Youn, S.-m. Ko, and T.-h. Kim, "Blockchain and ai meet in the metaverse," *Advances in the Convergence of Blockchain and Artificial Intelligence*, vol. 73, 2022.

[4] R. Casado-Vara, J. Prieto, F. De la Prieta, and J. M. Corchado, "How blockchain improves the supply chain: Case study alimentary supply chain," *Procedia computer science*, vol. 134, pp. 393–398, 2018.

[5] https://dappradar.com/.

[6] https://docs.bitdao.io/.

[7] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for ai: Review and open research challenges," *IEEE Access*, vol. 7, pp. 10 127–10 149, 2019.

[8] F. Chen, H. Wan, H. Cai, and G. Cheng, "Machine learning in/for blockchain: Future and challenges," *Canadian Journal of Statistics*, vol. 49, no. 4, pp. 1364–1382, 2021.

[9] H. Wang, C. Xu, C. Zhang, J. Xu, Z. Peng, and J. Pei, "vchain+: Optimizing verifiable blockchain boolean range queries," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1927–1940.

[10] C. Xu, C. Zhang, J. Xu, and J. Pei, "Slimchain: scaling blockchain transactions through off-chain storage and parallel processing," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2314–2326, 2021.

[11] C. Zhang, C. Xu, H. Wang, J. Xu, and B. Choi, "Authenticated keyword search in scalable hybrid-storage blockchains," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 996–1007.

[12] J. Eberhardt and S. Tai, "Zokrates-scalable privacy-preserving off-chain computations," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1084–1091.

[13] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *arXiv preprint arXiv:1908.04756*, 2019.

[14] R. K. Raman, R. Vaculin, M. Hind, S. L. Remy, E. K. Pissadaki, N. K. Bore, R. Daneshvar, B. Srivastava, and K. R. Varshney, "Trusted multi-party computation and verifiable simulations: A scalable blockchain approach," *arXiv preprint arXiv:1809.08438*, 2018.

[15] M. Fang, Z. Zhang, C. Jin, and A. Zhou, "High-performance smart contracts concurrent execution for permissioned blockchain using sgx," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1907–1912.

[16] S. Woo, J. Song, and S. Park, "A distributed oracle using intel sgx for blockchain-based iot applications," *Sensors*, vol. 20, no. 9, p. 2725, 2020.

[17] Z. Bao, Q. Wang, W. Shi, L. Wang, H. Lei, and B. Chen, "When blockchain meets sgx: An overview, challenges, and open issues," *IEEE Access*, vol. 8, pp. 170 404–170 420, 2020.

[18] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, "zkcrowd: a hybrid blockchain-based crowdsourcing platform," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4196–4205, 2019.

[19] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer pbft consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2020.

[20] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 423–443.

[21] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[22] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[23] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," *arXiv preprint arXiv:1810.13306*, 2018.

[24] K. Das and R. N. Behera, "A survey on machine learning: concept, algorithms and applications," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 2, pp. 1301–1309, 2017.

[25] S.-C. Wang, "Artificial neural network," in *Interdisciplinary computing in java programming*. Springer, 2003, pp. 81–100.

[26] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[27] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[28] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2003, pp. 986–996.

[29] T. Yu, Z. Lin, and Q. Tang, "Blockchain: The introduction and its application in financial accounting," *Journal of Corporate Accounting & Finance*, vol. 29, no. 4, pp. 37–47, 2018.

[30] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International journal of web and grid services*, vol. 14, no. 4, pp. 352–375, 2018.

[31] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[32] P. Vasin, "Blackcoin's proof-of-stake protocol v2," *URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf*, vol. 71, 2014.

[33] J. Deshpande, M. Gowda, M. Dixit, M. Khubbar, B. Jayasri, and S. Lokesh, "Permissioned blockchain based public procurement system," in *Journal of Physics: Conference Series*, vol. 1706, no. 1. IOP Publishing, 2020, p. 012157.

[34] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[35] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *2018 IEEE international conference on communications (ICC)*. IEEE, 2018, pp. 1–6.

[36] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3559–3570, 2019.

[37] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W.-C. Hong, "Machine learning adoption in blockchain-based smart applications: The challenges, and a way forward," *IEEE Access*, vol. 8, pp. 474–488, 2019.

[38] C. Li, B. Palanisamy, and R. Xu, "Scalable and privacy-preserving design of on/off-chain smart contracts," in *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2019, pp. 7–12.

[39] J. Eberhardt and J. Heiss, "Off-chaining models and approaches to off-chain computations," in *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2018, pp. 7–12.

[40] J. Eberhardt and S. Tai, "On or off the blockchain? insights on off-chaining computation and data," in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2017, pp. 3–15.

[41] J. D. Harris and B. Waggoner, "Decentralized and collaborative ai on blockchain," in *2019 IEEE international conference on blockchain (Blockchain)*. IEEE, 2019, pp. 368–375.

[42] J. Heiss, J. Eberhardt, and S. Tai, "From oracles to trustworthy data on-chaining systems," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 496–503.

[43] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*. Springer, 2010, pp. 465–482.

[44] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 626–645.

[45] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, 2018.

[46] A. Kosba, D. Papadopoulos, C. Papamanthou, and D. Song, "{MIRAGE}: Succinct arguments for randomized algorithms with applications to universal {zk-SNARKs}," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2129–2146.

[47] T. Schaffner, "Scaling public blockchains," *A comprehensive analysis of optimistic and zero-knowledge rollups. University of Basel*, 2021.

[48] G. Damaskinos, R. Guerraoui, R. Patra, M. Taziki *et al.*, "Asynchronous byzantine machine learning (the case of sgd)," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1145–1154.

[49] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[50] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[51] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[52] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[53] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[54] V. Trón, A. Fischer, D. A. Nagy, Z. Felföldi, and N. Johnson, "Swap, swear and swindle: incentive system for swarm," *Ethereum Orange Paper*, 2016.

[55] I. Storj Labs, "Storj: A decentralized cloud storage network framework," 2018.

[56] D. Vorick and L. Champine, "Sia: Simple decentralized storage," *Retrieved May*, vol. 8, p. 2018, 2014.

[57] J. Groth, "On the size of pairing-based non-interactive arguments," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.

[58] https://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples.

[59] https://archive.ics.uci.edu/ml/datasets/3D+Road+Network+(North+Jutland,+Denmark).

[60] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin concepts, threats, and machine-learning security solutions," *IEEE Access*, vol. 6, pp. 67 189–67 205, 2018.

[61] S. S. Arumugam, V. Umashankar, N. C. Narendra, R. Badrinath, A. P. Mujumdar, J. Holler, and A. Hernandez, "Iot enabled smart logistics using smart contracts," in *2018 8th International Conference on Logistics, Informatics and Service Sciences (LISS)*. IEEE, 2018, pp. 1–6.

[62] T. N. Dinh and M. T. Thai, "Ai and blockchain: A disruptive integration," *Computer*, vol. 51, no. 9, pp. 48–53, 2018.

[63] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *2018 IEEE international conference on big data (big data)*. IEEE, 2018, pp. 1178–1187.

[64] R. Doku and D. Rawat, "Pledge: A private ledger based decentralized data sharing framework," in *2019 Spring Simulation Conference (SpringSim)*. IEEE, 2019, pp. 1–11.

[65] P. Mamoshina, L. Ojomoko, Y. Yanovich, A. Ostrovski, A. Botezatu, P. Prikhodko, E. Izumchenko, A. Aliper, K. Romantsov, A. Zhebrak *et al.*, "Converging blockchain and next-generation artificial intelligence technologies to decentralize and accelerate biomedical research and healthcare," *Oncotarget*, vol. 9, no. 5, p. 5665, 2018.

[66] E. C. Ferrer, O. Rudovic, T. Hardjono, and A. Pentland, "Robochain: A secure data-sharing framework for human-robot interaction," *arXiv preprint arXiv:1802.04480*, 2018.

[67] T.-T. Kuo and L. Ohno-Machado, "Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks," *arXiv preprint arXiv:1802.01746*, 2018.

[68] J. Eberhardt, "Scalable and privacy-preserving off-chain computations," 2021.

[69] B. Gu, Z. Li, A. Liu, J. Xu, L. Zhao, and X. Zhou, "Improving the quality of web-based data imputation with crowd intervention," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2534–2547, 2019.

[70] X. L. Dong, B. Saha, and D. Srivastava, "Less is more: Selecting sources wisely for integration," *Proceedings of the VLDB Endowment*, vol. 6, no. 2, pp. 37–48, 2012.

[71] C. Chai, J. Fan, and G. Li, "Incentive-based entity collection using crowdsourcing," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 341–352.

[72] Y. Wang, Z. Cai, Z.-H. Zhan, Y.-J. Gong, and X. Tong, "An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 414–429, 2019.

[73] R. Kunkel, D. L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "Tensorscone: A secure tensorflow framework using intel sgx," *arXiv preprint arXiv:1902.04413*, 2019.

[74] Y. Sun, S. Wang, H. Li, and F. Li, "Building enclave-native storage engines for practical encrypted databases," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 1019–1032, 2021.

[75] M. Zhao, M. Gao, and C. Kozyrakis, "Shef: Shielded enclaves for cloud fpgas," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1070–1085.

[76] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. D. Ciccio, I. Weber, M. Wöhrer, and U. Zdun, "Foundational oracle patterns: Connecting blockchain to the off-chain world," in *International Conference on Business Process Management*. Springer, 2020, pp. 35–51.

[77] A. Beniiche, "A study of blockchain oracles," *arXiv preprint arXiv:2004.07140*, 2020.

[78] G. Caldarelli, "Understanding the blockchain oracle problem: A call for action," *Information*, vol. 11, no. 11, p. 509, 2020.

[79] J. Stark, "Making sense of ethereum's layer 2 scaling solutions: State channels, plasma, and truebit," *Medium. com*, 2018.

[80] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[81] A. Garoffolo and R. Viglione, "Sidechains: Decoupled consensus between chains," *arXiv preprint arXiv:1812.05441*, 2018.

[82] A. Miller, I. Bentov, R. Kumaresan, and P. McCorry, "Sprites: Payment channels that go faster than lightning," *CoRR, abs/1702.05812*, 2017.

[83] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," *arXiv preprint arXiv:2107.10881*, 2021.

[84] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Blockchain and machine learning for communications and networking systems," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1392–1431, 2020.

[85] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[86] C. Xu, K. Wang, and M. Guo, "Intelligent resource management in blockchain-based cloud datacenters," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 50–59, 2017.

[87] S. Fei, Z. Yan, W. Ding, and H. Xie, "Security vulnerabilities of sgx and countermeasures: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[88] B. Podgorelec, M. Turkanović, and S. Karakatič, "A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection," *Sensors*, vol. 20, no. 1, p. 147, 2019.

[89] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: a review," *Ieee Access*, vol. 6, pp. 10 179–10 188, 2018.

[90] J. Zhang, Z. Fang, Y. Zhang, and D. Song, "Zero knowledge proofs for decision tree predictions and accuracy," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 2039–2053.

[91] L. Zhao, Q. Wang, C. Wang, Q. Li, C. Shen, and B. Feng, "Veriml: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2524–2540, 2021.

[92] H. Wu, W. Zheng, A. Chiesa, R. A. Popa, and I. Stoica, "{DIZK}: A distributed zero knowledge proof system," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 675–692.

[93] S. Setty, S. Angel, T. Gupta, and J. Lee, "Proving the correct execution of concurrent services in zero-knowledge," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 339–356.

[94] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 253–270.

[95] "Zcash company," https://z.cash/.

[96] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.

[97] G. Zyskind, O. Nathan, and A. Pentland, “Enigma: Decentralized computation platform with guaranteed privacy,” *arXiv preprint arXiv:1506.03471*, 2015.

[98] Y. Zhu, X. Song, S. Yang, Y. Qin, and Q. Zhou, “Secure smart contract system built on smpc over blockchain,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1539–1544.

[99] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[100] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, “Federated learning for internet of things: Recent advances, taxonomy, and open challenges,” *IEEE Communications Surveys & Tutorials*, 2021.

[101] C. Gentry, *A fully homomorphic encryption scheme*. Stanford university, 2009.

[102] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Scalable zero knowledge via cycles of elliptic curves,” *Algorithmica*, vol. 79, no. 4, pp. 1102–1160, 2017.

[103] S. Disabato and M. Roveri, “Incremental on-device tiny machine learning,” in *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, 2020, pp. 7–13.

[104] B. Gu, S. Kargar, and F. Nawab, “Efficient dynamic clustering: Capturing patterns from historical cluster evolution,” *arXiv preprint arXiv:2203.00812*, 2022.

[105] Ó. Fontenla-Romero, B. Guijarro-Berdiñas, D. Martinez-Rego, B. Pérez-Sánchez, and D. Peteiro-Barral, “Online machine learning,” in *Efficiency and Scalability Methods for Computational Intellect*. IGI Global, 2013, pp. 27–54.

[106] S. C. Hoi, J. Wang, and P. Zhao, “Libol: A library for online learning algorithms,” *Journal of Machine Learning Research*, vol. 15, no. 1, p. 495, 2014.