National Research Journal of Information Technology & Information Science
Volume No: 11, Issue No: 1, Year: 2024 (January-June)
PP: 126-137
ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

REVOLUTIONIZING MACHINE LEARNING THROUGH BLOCKCHAIN: AN OPTIMISTIC PERSPECTIVE

Dinesh Yadav

M.Tech, Student WCTM

Vishali Sharma

Assistant Professor, WCTM

ABSTRACT

The convergence of blockchain technology with machine learning has garnered notable attention, spurred by the vision of decentralized, secure, and transparent AIservices. Within this landscape, we present Chain learn (Blockchain-Powered Optimistic Machine Learning), a novel methodology that empowers blockchain frameworks to perform AI model inference. Chain learn employs an interactive fraud-proof protocol, reminiscent of optimistic rollup systems, to ensure decentralized and verifiable consensus for machine learning services, thereby enhancing trust and transparency. In contrast to zkML (Zero-Knowledge Machine Learning), Chain learn offers cost-effective and highly efficient ML services with minimal participation requirements. Notably, Chain learn facilitates the execution ofextensive language models, such as 7B-LLaMA, on standard PCs without GPUs, substantially broadening accessibility. Through the amalgamation of blockchain andAI capabilities via Chain learn, we embark on a transformative journey towards accessible, secure, and efficient on-chain machine learning.

Keywords: Blockchain, Machine Learning, Fraud Proof, Rollup.

1. INTRODUCTION

In the dynamic landscape of digital advancement, the fusion of Artificial Intelligence (AI) and blockchain technology signifies a transformative shift in how we engage with and leverage information. AI, known for its advanced data analysis and decision- making abilities, and blockchain, a decentralized ledger celebrated for its security and transparency, have united to pioneer novel avenues in the digital domain. As distinct forces with unique capabilities, the convergence of AI and blockchain is redefining the boundaries of digital possibility. This convergence has introduced the concept of "Onchain AI," a cutting-edge paradigm prepared to provide decentralized, safe, and effective AI services directly within the blockchainnetwork. However, a prevalent challenge within the realm of "Onchain AI" is the impracticality of executing AI computations directly on the blockchain. For example, a seemingly simple task like matrix multiplication of 1000 × 1000 integers would incur over 3 billion gas costs, far surpassing the current gas limit imposed byEthereum. As a result, many of these applications opt for off-chain computations on centralized servers, only transferring the outcomes onto the blockchain. While this approach may yield functional outputs, it inherently compromises decentralization. This compromise not only introduces significant security concerns but also undermines the fundamental principles of trust and transparency that blockchain technology seeks to promote. An alternative approach involves harnessing Zero- Knowledge Machine Learning (zkML), representing a groundbreaking paradigm in merging machine learning and blockchain technologies. zkML's utilization of zk- SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) has played a crucial role in safeguarding sensitive model parameters and user data throughout the training and inference phases. This not only addresses privacy

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

concerns but also alleviates the computational strain on the blockchain network, positioning zkML as a promising solution for decentralized machine learning applications. While zkML offers undeniable advantages in bolstering privacy and security within machine learning on the blockchain, it's vital to recognize its inherent limitations. One of the most notable challenges lies in the substantial cost associated with proof generation in zkML.

The process necessitates significant computational resources, leading to prolonged generation times and extensive memory usage. Consequently, zkML is most suitable for relatively modest models, as its inefficiency becomes evident when handling larger and more intricate models. For instance, zkML would require over 1000 times the memory and computational resources for ZK proof generation. As a result, zkML may prove impractical for expansive AI applications necessitating the handling of large-scale datasets and complex model parameters. In response to the limitations of zkML, we explore the utilization of fraud proof to validate the accuracy of ML outcomes on the blockchain, as opposed to relying on zero-knowledge proofs (ZKPs), also known as validity proofs. Fraud proofs are a common feature in blockchain systems, particularly within rollup systems, which belong to the broader category of optimistic systems. Prominent examples of rollup systems utilizing Arbitrum and Optimism as fraud evidence. In a system employing fraud proof, there's an optimistic assumption that each proposed result is valid by default. However, in cases where there's a suspicionof invalid results, the system introduces a challenge period during which participants can challenge the submitter. The fraud proof is generated through an interactive pinpoint protocol, demonstrating that the provided result is erroneous. The arbitration process is designed to validate a fraud proof with minimal computational steps, ensuring that the on-chain cost remains exceedingly low. Encouraging system design as a basis, we provide Chainlearn: Optimistic Machine Learning on the blockchain1. Diverging from the approach of zkML, which relies on zero- knowledge proofs, Chainlearn adopts a fraud-proof system to guarantee the correctness of ML results. Submitters can run machine learning (ML) algorithms in their own environment and then publish the results straight to the blockchain using the Chainlearn platform. This approach maintains an optimistic assumption that each proposed result is inherently valid. During the challenge process, validators will check the correctness of these submitted results. If the results are invalid, he will start the dispute game (bisection protocol) with the submitter and tries to disprove the claim by pinpointing one concrete erroneous step.

2. DESIGN PRINCIPLES

The design principles underlying ChainLearn are outlined as follows:

1. **Deterministic ML Execution:** To ensure consistency and determinism in ML execution, ChainLearn employs fixed-point arithmetic and software- based floating-point libraries. This approach guarantees that the ML execution process can be represented by a deterministic state transition function, mitigating variability stemming from randomness and floating- point computation.
2. **Separate Execution from Proving:** ChainLearn utilizes a dual-target compilation approach, compiling the same source code twice. One compilation is optimized for native execution, leveraging multithreaded CPUs and GPUs for enhanced speed. The other compilation targets fraud- proof VM instructions for use in the fraud-proof protocol. This strategy ensures swift execution while maintaining machine-independent code for proving.
3. **Optimistic Machine Learning with Interactive Fraud Proofs:** Adoptinginteractive fraud proof mechanisms, ChainLearn employs a process that progressively resolves

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

disputes to a single instruction level, ultimately resolving base-case discrepancies using on-chain fraud-proof VMs.

4. **Optimizing ML Fraud Proof with Multi-Phase Protocol:** Traditional fraud-proof systems, as prevalent in optimistic rollup systems, often require cross-compilation of entire computations into fraud-proof VM instructions, leading to inefficient execution and substantial memory consumption. ChainLearn introduces a novel multi-phase protocol featuring semi-native execution and lazy loading. This approach significantly accelerates the fraud-proof process while minimizing memory overhead.

3. ARCHITECTURE

ChainLearn relies on a fraud-proof mechanism to verify the accuracy of machine learning outcomes on the blockchain. This fraud-proof mechanism within ChainLearn comprises three essential elements:

1. Fraud-Proof Virtual Machine (FPVM): FPVM can track any instruction step of a stateless program along with its inputs and validate it on the layer 1blockchain (L1).
2. Machine Learning Engine: This highly efficient engine is tailored to handleboth native execution and fraud-proof scenarios. It ensures fast and precise execution of machine learning tasks while maintaining result consistency and determinism.
3. Interactive Dispute Resolution: The dispute resolution process in ChainLearn breaks down disputes to single instructions and resolves them using on-chain FPVM.

4. WORKFLOW:

On-chain verification is crucial in ChainLearn. Verifiers, also known as challengers, scrutinize these outcomes. In case of disputes, the bisection protocol is activated to pinpoint and rectify any identified errors. The workflow in ChainLearnunfolds as follows:

1. The requester initiates an ML service task.
2. The submitter executes the ML service task and records the result on-chain.
3. Verifiers (challengers) validate the results. If deemed incorrect, a verifier initiates the dispute game (bisection protocol) with the server to disprove the claim by identifying specific erroneous steps.
4. Smart contract arbitration resolves the disputed step, offering a conclusiveresolution to the dispute.
5. After a specified "challenge period," the results are confirmed.

Both the server (submitter) and the verifier (challenger) are required to stake in the system. Providing incorrect results results in a loss of stake. Thus, if all parties follow their incentives, only valid results are committed. Consequently, in most cases, the server (submitter) provides correct results, and verifiers (challengers) validate them, minimizing the occurrence of the dispute game. After the defined "challenge period," the results are confirmed.

5. FRAUD PROOF VIRTUAL MACHINE

We've developed a Fraud Proof Virtual Machine (FPVM) for executing off-chain and handling on-chain arbitration. We ensure that the off-chain VM is equivalent to the on-chain VM implemented in a smart contract. Essentially, the FPVM operates as a state transition function where each operation, termed as a Step, executes a single instruction. Given an input

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

state $Spre$ with instructions encoded within it, the FPVM, acting as a state transition function denoted as $(\cdot)$, generates a new state $Spost$, represented as $VM(Spre) \rightarrow Spost$. As a result, an ordered set of VM states makes up the trace of a program run by the FPVM, whichshows the results of running a machine learning program on the FPVM.

The execution trace T is a sequence ($S0$, $S1$, $S2$, $\cdots$, Sn), where each Si represents a VM state, and $Si = VM(Si-1)$, $\forall i \in [1, n]$. Every execution trace has a unique initial state $S0$, determined by the ML model and input.

Management of the VM state is facilitated by a Merkle tree, where only the Merkle root is uploaded to the on-chain smart contract, representing the VM state. The memory layout in the FPVM comprises various areas, including "program code," "input," "output," "oracle key," "oracle value," and "model," as depicted in figure below. The machine learning program resides in the "program" field, the input for the ML model is stored in the "input" field, and the ML model itself is situated in the "model" field. The output of chainlearn is placed in the "output" field upon completion of the ML program execution. Furthermore, the fraud-proof system hasa key-value oracle that the FPVM uses to access outside data as it changes states.

The oracle key and value are stored in the "oracle key" and "oracle value" fields, respectively. Furthermore, the FPVM memory is structured as a Merkle tree with a fixed depth of 27 levels, with leaf values comprising 32 bytes each. This structure covers the full 32-bit address space, where each leaf contains the memory data. The Merkle root of the tree reflects the effects of memory writes in the FPVM, enabling representation of each field in the FPVM memory as a Merkle subtree root.

6. MACHINE LEARNING ENGINE

Within ChainLearn, we've developed a highly efficient machine learning engine to accommodate both native execution and fraud-proof scenarios. This engine not only facilitates rapid and precise execution of machine learning tasks but also ensures the consistency and determinism of results. This aspect is especially vital during dispute resolution, as the machine learning engine can reliably produce a valid output and verify disputes in a stateless manner, thereby bolstering thedependability of the entire system.

7. SEPARATE EXECUTION FROM PROVING

Following the design principle of "Separate Execution from Proving," we have designed a highly efficient machine learning engine for chainLearn. This engine offers two types of implementations: one optimized for native execution, prioritizing speed, and utilizing multi-threaded CPUs and GPUs for acceleration;the other compiled into a fraud-proof program for FPVM. This dual-target approach ensures swift execution while maintaining the integrity of the proving process, which relies on machine-independent code. Consider the example of matrix multiplication within the machine learning engine,as depicted in code below. During native execution, GPU calculation (using CUDA) is employed for acceleration, as illustrated in code below. However, because CUDA calculator compatibility is not supported, the machine learning engine is compiled into machine-independent FPVM instructions for the proof step. Both implementations guarantee consistent execution results across different scenarios.

// a,b,c are GPU device pointers to matrix void GpuMatrixMultiplication(int *a, int *b, int *c, int m, int n, int k) { int row = blockIdx.y * blockDim.y + threadIdx.y; int col = blockIdx.x * blockDim.x + threadIdx.x; int sum = 0; for(int i = 0; i < n; i++) {sum += a[row * n + i] * b[i * k + col];} c[row * k + col] = sum; The process of matrix multiplication within the machine learning engine, utilizing GPU acceleration for native execution

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

In our current implementation, we've prioritized the efficiency of AI model computation within the FPVM. To achieve this, we've developed a lightweight machine learning engine tailored specifically for this purpose, rather than relying on popular ML frameworks like TensorFlow or PyTorch. Additionally, we providea script capable of converting models from TensorFlow and PyTorch to this lightweight library. With this script, models trained on TensorFlow or PyTorch canbe converted to the ChainLearn model format. Consistency and determinism are crucial aspects addressed in our implementation. Randomness and unpredictability in floating-point calculations are the two main causes of inconsistencies in machine learning outcomes.

To mitigate randomness, it's common practice to fix the random seed in the random number generator, as computer-generated randomness is essentially pseudo-random. Addressing inconsistencies in floating-point computations is more complex Variations in execution results may arise from floating-point number properties during native DNN computations, particularly across multiple hardware platforms. Rounding mistakes can lead to non-identical results in parallel computations involving floating-point numbers, such as $(a\ b)\ c$ vs $a\ (b\ c)$. Additionally, variables like as operating system, compiler version, and programming language can further affect how floating-point numbers are computed, which exacerbates inconsistent ML results.

To address these challenges and ensure the consistency of ChainLearn, we employ two key approaches:

1. **Fixed-Point Arithmetic:** We adopt fixed-point arithmetic, also known as quantization technology. By employing this method, we can represent and execute calculations with fixed precision instead of floating-point integers.

   By doing so, we mitigate the effects of floating-point rounding errors, resulting in more reliable and consistent results. It's important to note that using fixed-point arithmetic may lead to a slight loss of accuracy in DNN models. This tradeoff between execution performance and model accuracy isan essential consideration when implementing such precision techniques.

2. **Software-Based Floating-Point Libraries (softfloat):** We utilize software- based floating-point libraries (softfloat) designed to operate consistently across different platforms. These libraries guarantee determinism and cross- platform consistency of the ML results, independent of the underlying software or hardware configurations.

   By integrating fixed-point arithmetic and software-based floating-point libraries (softfloat), we establish a robust foundation for achieving consistent and reliable ML results within the ChainLearn framework.

8. LOCATING THE DISPUTED POINT:

At the outset, the submitter and verifier agree on the initial state $S0$ but disagree onthe final state $Sn \neq Sn'$. The objective of the protocol, depicted in Figure 5, is to pinpoint a specific VMk within a sequence of VM instructions executed within thecontext of the initial VM state $S0$ (with n such instructions in total), where $Sk-1 \neq$

$Sk-1$ but $Sk = Sk'$.

The dispute game unfolds in a sequence of rounds. At the start of each round, the submitter and verifier agree on a starting VM state Si and disagree on the endingstate $Si+j$ for some $j >$ 1. At first, $i = 0$ and $j = n$.

. Subsequently, the challenger must claim the state Sm at the midpoint of the VMstate, where

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

$m = i + \lfloor j/2 \rfloor$. Then, the submitter confirms or disagrees with the challenger's midpoint state Sm.

Two scenarios can unfold:

1. If the submitter agrees with the challenger's midpoint state, a smaller dispute is identified, and the protocol proceeds with the next round, adjusting i and j accordingly.
2. If the submitter disagrees with the challenger's midpoint state, another smaller dispute is identified, and the protocol proceeds to the next round.

In both cases, the length of the dispute is halved, and the procedure is repeated until $j = 1$. The efficiency of the dispute game is noteworthy. In terms of time complexity, both the verifier and submitter require only $\lceil \log n \rceil$ rounds of challenge-response to converge on the same value of k, where $Sk-1 \neq Sk-1$ but Sk

$= Sk'$. A timeout penalty-equipped challenge-response mechanism helps to achieve this synchronization.

Subsequently, the state $Sk-1$, along with auxiliary data, is forwarded to a smart contract for arbitration.

9. ONCHAIN ARBITRATION

For on-chain arbitration, the submitter and verifier will send (VMIk, S$k-1$, Sk) tothe contract for arbitration. The on-chain VM will take S$k-1$ as input and conducta one-step execution to output the correct S. Because the on-chain virtual machine (VM) only performs a single instruction, the amount of data (witness) that must beretrieved is a mere 0(1).

In on-chain arbitration, the witness comprises a partial expansion of the Merkle tree representing the before state $Sk-1$. The on-chain VM uses this partially expanded state tree to read the next instruction, emulate the instruction execution, and then compute the Merkle root hash of the resulting state. Notably, the one-stepon-chain VM execution always requires only (1) computation and memory consumption, ensuring that on-chain arbitration can be conducted using a feasibleamount of Ethereum gas.

The challenger wins if they are able to present a legitimate one-step evidence at theon-chain arbitration.

. Otherwise, the submitter wins the challenge.

10. MULTI-PHASE DISPUTE GAME

10.1 LIMITATIONS OF ONE-PHASE DISPUTE GAME

With the design principle of "Separate Execution from Proving," we can achieve high performance in the optimistic scenario where the submitter consistently provides correct results. However, to safeguard against malicious behavior in the pessimistic scenario, we still require generating a fraud-proof for Chainlearn. In chainlearn's fraud-proof protocol, we cross-compile the ML computation into fraud-proof VM instructions and then initiate the dispute game to identify the disputed step. However, this approach of cross-compiling the entire ML computation into fraud-proof VM instructions presents significant limitations:

1. **Low Execution Efficiency:** The one-phase dispute game suffers from a critical drawback: for proving, all computations must be executed within the FPVM, thereby preventing us from harnessing the full potential of GPU/TPU acceleration or parallel processing. This constraint thus seriously compromises the effectiveness of offering

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

fraud-proofing for large-scale model inference, and it is also consistent with the existing restriction of the referred delegation of the computation (RDoC) protocol.

2. **Restricted Memory in Fraud-Proof VM:** The MIPS VM, for instance, can only accommodate up to 4 GB of memory. The fraud-proof VM has limited memory capacity.

Due to this limitation, we cannot directly load a large model into the fraud-proof VM. For instance, the size of a 7B-llama model in float64 is around 26GB, exceeding the memory capacity of the MIPS FPVM.

10.2 OVERVIEW OF MULTI-PHASE PROTOCOL

To overcome the constraints of the one-phase protocol and ensure that chainlearn can generate fraud-proof with performance comparable to the native environment,we introduce a multi-phase protocol. The multi-phase dispute game offers the following features to effectively address the aforementioned limitations:

1. Semi-Native Execution: By using a multi-phase architecture, we reduce the amount of processing required in the virtual machine (VM) until the last stage,which is similar to a single-phase protocol.
2. During earlier phases, we have the flexibility to execute computations leading to state transitions in the native environment, leveraging parallel processing capabilities in CPU, GPU, or even TPU. By reducing reliance on the VM, we significantly decrease overhead, resulting in a notable improvement in chainlearn's execution performance, almost matching that of the native environment.
3. Lazy Loading Design: To optimize memory usage and VM performance, we implement a lazy loading technique. Instead of loading all data into the VM memory at once, we only load keys identifying each data item. When the VM needs to access a specific data item, it retrieves it from an external source using the key and loads it into memory. Once the data item is no longer required, it is swapped out of memory to free up space for other data items. This approach allowsus to handle large datasets without exceeding memory capacity or compromising VM efficiency.
4. Figure below illustrates a verification game comprising two phases (k = 2). Phase-2 procedures are similar to those of a single-phase verification game in that every change in state is attributed to a single VM micro-instruction that modifies the VM state.
5. Phase-1 state transitions are equivalent to a "Large Instruction" consisting of several micro-instructions that alter the context of processing.
6. . Initially, the submitter and challenger engage in the dispute game during Phase-1 using a bisection protocol to pinpoint the dispute step within a "large instruction."This step is then forwarded to Phase-2. Phase-2 operates similarly to the single- phase dispute game. The bisection protocol in Phase-2 aids in identifying the dispute step within a VM micro-instruction. Subsequently, this step is sent to the arbitration contract on the blockchain.

10.3 STATE TRANSITION TO NEXT PHASE:

To maintain the integrity and security of the transition to the next phase, we utilize the Merkle tree. Illustrated in Figure 7, this process entails extracting a Merkle sub- tree reconstruction, ensuring the smooth continuation of the dispute game process. As an

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

example, let's look at the state transition in the two-phase protocol.

In Phase-1, after the submitter and challenger have identified the dispute step on a "large instruction", denoted as $Si-1 \neq Si$, we proceed to construct a Merkle tree on the state data $Si-1$ to obtain the Merkle tree root $ro(Si-1)$. Next, we initialize the initial state in the next phase, denoted as $Sk+1$, using this Merkle root.

Specifically, we start with an empty VM image, set up the running program in the "program code" memory field within the FPVM, and designate the Merkle root

$ro(Si-1)$ as a key for lazy loading in the "input" memory field of the FPVM. Once initialization and data filling are complete, we obtain the initial state $Sk+1$ for the next phase. Due to the complexity of constructing $Sk+1$ and the gas limitations of Ethereum, we employ a zero-knowledge circuit to verify the correctness of $Sk+1$'s construction and validate the zero-knowledge proof on the chain

Similarly, at the conclusion of the execution in Phase-2, we need to verify the consistency between the submitter's state Si and the challenger's state Si'.In particular, the state Si is stored in the "output" field of the FPVM memory, and we can verify its integrity by providing a Merkle proof.

10.4 DNN COMPUTATION IN MULTI-PHASE CHAINLEARN:

In this demonstration of a two-phase Chainlearn approach, we focus on the computation process of Deep Neural Networks (DNNs) represented as a computation graph, denoted as G. This is an outline of the steps involved in theprocess:

1. **Computation Graph Representation**:

   The DNN computation is visualized as a computation graph, G, consisting of various nodes representing different computational steps. These nodes store intermediate computation results as the graph progresses.

2. **DNN Model Inference**:

   DNN model inference involves executing computations on the computation graph, G. Initially, the entire graph serves as the inference state, constituting the computation context in Phase-1. As computations are performed on individual nodes, the graph advancesto its subsequent state.

3. **Phase-1 Dispute Game**:

   In Phase-1, the dispute game is conducted directly on the computation graph. Computation on graph nodes can be carried out in a native environment using multi-thread CPU or GPU for efficiency. The bisection protocol assists in identifying the disputed node, whose computation will be transitioned to Phase-2 for further resolution.

4. **Phase-2 Bisection**:

   Transitioning to Phase-2, the computation of the disputed node is transformed into Virtual Machine (VM) instructions, akin to the single-phase protocol. This allows for a structured resolution process to address any discrepancies or disputes identified in Phase-1.

By implementing a two-phase Chainlearn approach, we streamline the resolution of disputes in DNN computations while optimizing efficiency. Additionally, we anticipate extending this approach to include more than two phases when dealingwith computationally complex nodes,

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

further enhancing the robustness and effectiveness of the fraud-proof protocol.

10.5 PERFORMANCE IMPROVEMENT:

In this section, we delve into a concise discussion and analysis of our proposed multi-phase fraud-proof framework. Let's consider the following scenario:

Assuming a DNN computation graph with n nodes, where each node requires m VM micro-instructions to complete the computation in the VM. Let α represent the speedup ratio achieved through GPU or parallel computing, which can range from

tens to hundreds of times faster than single-thread VM execution. These premises allow us to arrive to the following conclusions:

1. **Performance**

   Two-phase Chainlearn surpasses single-phase Chainlearn in terms of performance, achieving a computation speedup of α times. By employing multi-phase verification, we capitalize on the accelerated computation capabilities offered by GPU or parallel processing, resulting in significant enhancements in overall performance.

2. **Space Complexity**

   Two-phase Chainlearn minimizes the space complexity of the Merkle tree. A comparison of Merkle tree space complexity reveals that in two-phase Chainlearn, the size is $(m + n)$, whereas in single-phase Chainlearn, the space complexity is substantially larger at (mn). This reduction in Merkle tree size underscores the efficiency and scalability of the multi-phase design.

In summary, the multi-phase fraud-proof framework presents a notable improvement in performance, ensuring more efficient and expedited computations, especially when leveraging the speedup capabilities of GPU or parallel processing. Furthermore, the decreased Merkle tree size enhances the system's effectiveness and scalability, positioning multi-phase Chainlearn as a compelling option for various applications.

11. SECURITY ANALYSIS:

Here, we perform a security analysis of our system using, for simplicity, Arbitrum's AnyTrust assumption.

11.1 AnyTrust Assumption:

The AnyTrust assumption posits that for every claim made, there exists at least one honest node. This implies that either the submitter is honest, or at least one verifier is honest and will challenge within the pre-defined period. Even if $m - 1$ verifiers collude to remain silent about a submitter's incorrect claim, the presence of an honest verifier ensures that the wrong claim can be disproven in front of the smart contract, resulting in its rejection. We also assume data availability and anti- censorship, which are addressed by orthogonal countermeasures.

11.2 Safety and Liveness:

Under the AnyTrust assumption, Chainlearn can maintain both safety and liveness.

1. **Safety**: A single honest validator can compel Chainlearn to behave correctly. If all nodes except one are malicious and one provides an incorrect result on- chain, the honest node will identify the error and initiate a challenge. Through the dispute game, the honest node and the malicious node will pinpoint the erroneous step, leading to the rejection of the incorrect result and penalization of the malicious node.

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

2. **Liveness**: Any proposed result will be either accepted or rejected by the contract within a maximum time period. The finite instruction set and execution traces of Chainlearn ensure that computations are completed within a maximum time Te. Even in the worst-case scenario where malicious verifiers attempt to delay the process, the result will be acceptedor rejected within a bounded time frame.

12. INCENTIVE MECHANISM

To ensure the safety and liveness of Chainlearn, it's crucial to incentivize validators to consistently verify results and encourage submitters to refrain from cheating.

This requires designing an incentive-compatible mechanism that aligns the interests of all participants.

Rational validators should be motivated to verify results diligently, knowing that atleast one honest validator will scrutinize the outcome. Similarly, rational submitters should have no incentive to cheat, understanding that any dishonest behavior will be swiftly detected and penalized.

By creating such an incentive structure, we can foster a trustworthy environment where all participants are incentivized to act honestly, contributing to the overall integrity and effectiveness of the Chainlearn system.

13. VERIFIER DILEMMA

In the Verifier's Dilemma scenario in Chainlearn, the following payoff matrix describes the potential outcomes for both the submitter and validators:

In this matrix:

1. S represents the stake placed by each participant on the chain.
2. C is the computation cost for Chainlarn execution.
3. B is the benefit obtained by the submitter from cheating.
4. L is the loss suffered by validators if the submitter cheats and isn'tchallenged.
5. R is the reward received by validators if the submitter is successfullychallenged.

The Nash equilibrium strategy in this verification game is one where neither the submitter nor the validators have an incentive to deviate from their chosen action, given the action of the other party. It typically occurs when both parties make decisions that maximize their own payoff, considering the decisions of the other party.

In the context of Chainlearn, achieving a Nash equilibrium strategy is crucial to ensure that both submitters and validators act in a manner that promotes the integrity and security of the system.

13.1 Attention Challenge

To address the Verifier's Dilemma in Chainlearn, we introduce the Attention Challenge mechanism. In this mechanism, instead of directly revealing the Chainlearn result $f(x)$ on chain, the submitter first reveals the hash of the result $H(As,f(x))$, where As is the address of the submitter and $H(\cdot)$ is a hash function.

Here's how the Attention Challenge mechanism works:

1. The submitter reveals the hash of the result on chain: $H(As,f(x))$.
2. Validators have a window of time to respond on chain with their own hash value

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

$H(Av,f(x))$, where Av is the address of the validator.

3. If $H(Av,f(x))<T$ for any validator, they must respond on chain within the timewindow.
4. After the time window expires, the submitter can post the actual chainlearnresult $f(x)$ on chain.
5. If any validator responded incorrectly or did not respond within the time window, the submitter can accuse them on chain. This accusation will be checked on chain, and if valid, the accused validator will be penalized, withhalf of the penalty going to the submitter and the other half being burned.

Theorem: If $pt \cdot G>C$, where pt is the probability that a validator needs to respond, G is the penalty for incorrect or non-response, and C is the computation cost, thenrational validators will always check, and rational submitters will never cheat.

Theorem 8.2. When $p_t \cdot G > C$, the only Nash equilibrium of the verification gamewith attention challenge mechanism is that validator will always check and the submitter will never cheat.

Proof : The utility for the validator to check the results is U (check) = $p_C R - C$,and the utility for the validator to be lazy and not check the results is U (lazy) = $-p_C \cdot L - p_t \cdot G$. When $p_t \cdot G > C$, we have that U (check) − U (lazy) = $p_C (R + C) + p_t G - C > 0, \quad \forall p_C \in [0, 1]$

Indeed, the proof is now complete. Regardless of the probability that the submittermay cheat, the dominant strategy for the validator remains to always check.

To be more precise, during the inference stage of a deep neural network (DNN) model, a simple forward computation is performed on the DNN computation graph, or G.

Conversely, the training process encompasses both forward computation andbackward update (backpropagation) on the same DNN computation graph G.Despite their differing objectives, the computation processes for forward

computation and backward update share similarities, allowing for a unified approach to handling both tasks within the chainlearn framework.

Through the incorporation of a multi-phase chainlearn approach, we can efficiently extend support to the training process. Here's how it operates: During each iterationof the training process, the dispute game is initiated to identify any disputes within that specific iteration. Subsequently, the process progresses to the next phase, where both the submitter and challenger participate in a dispute protocol concerning the computation graph for both forward and backward processes. This facilitates the identification of the disputed node, and the computation associated with this node is then forwarded to the subsequent phase, where the fraud-proof VM arbitrates the issue.

Verifying ML model development on the blockchain is significantly impacted by the expansion of chainlearn to include the training phase.

By leveraging on-chain data for training and updating the ML model, chainlearn ensures auditability and transparency throughout the proce Knowing this, the dominant strategy for the submitter becomes clear: always provide a correct result. This strategic alignment ensures that both parties act in their best interest, leading to a system where honesty is incentivized and cheatingis disincentivized. With this, the security and reliability of the chainlearn system are effectively ensured.

The cost analysis demonstrates that the introduction of the attention challenge mechanism in chainlearn incurs minimal expenses. Assuming an interest rate r for locking up the stake G

ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

and a gas fee t for the validator to respond, the extra cost forimplementing this mechanism in chainlearn amounts to $rG + tpt$. By setting $G = \sqrt{(tC/r)}$ and $pt = \sqrt{(rC/t)}$, the minimal cost is at least $2\sqrt{(rtC)}$.

For instance, considering the specific case of a GPT-3.5 inference taking 1 second on an A100 with a raw computation cost of \$0.001 for 1,000 tokens, with r = 0.001and t = \$1, setting G = 1 and pt = 0.1%, the total cost for implementing the attention challenge per chainlearn request amounts to only \$0.002. This minimal cost underscores the affordability and feasibility of integrating the attention challenge mechanism into chainlearn.

14. CONCLUSION

In conclusion, the optimistic approach to on-chain AI and machine learning presents significant advantages over existing methods, allowing for the integration of AI capabilities with the integrity and security inherent in blockchain technology. Chainlearn emerges as a standout solution, offering a cost- effective and efficient ML service when compared to zkML. This positions chainlearn as a crucial player in shaping the future of decentralized, secure, and transparent AI services. As the landscape of on-chain AI continues to evolve, chainlearn remains a pivotal solution, unlocking the full potential of these technologies and spearheading a transformative journey towards accessible, secure, and efficient on-chain machine learning.

REFERENCES

Ben Goertzel, Simone Giacomelli, David Hanson, Cassio Pennachin, and Marco Argentieri. 2017. SingularityNET: A decentralized, open market and inter- network for AIs. Thoughts, Theories Stud. Artif. Intell. Res. (2017).

Justin D Harris and Bo Waggoner. 2019. Decentralized and collaborative AI on blockchain. In 2019 IEEE international conference on blockchain (Blockchain).

Karl Wüst, Sinisa Matetic, Silvan Egli, Kari Kostiainen, and Srdjan Capkun. 2020. ACE: Asynchronous and concurrent execution of complex smart contracts. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 587–600.

Ran Canetti, Ben Riva, and Guy N Rothblum. 2011. Practical delegation of computation using multiple servers. In Proceedings of the 18th ACM conference on Computer and communications security. 445–454.

Thang N Dinh and My T Thai. 2018. AI and blockchain: A disruptive integration. Computer 51, 9 (2018), 48–53.

Tianyi Liu, Xiang Xie, and Yupeng Zhang. 2021. ZkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2968–2985.

Z Chen, W Wang, X Yan, and J Tian. 2018. Cortex-AI on blockchain: The decentralized AI autonomous system. Cortex White Paper (2018).

Zhibo Xing, Zijian Zhang, Meng Li, Jiamou Liu, Liehuang Zhu, Giovanni Russello, and Muhammad Rizwan Asghar. Zero-Knowledge Proof-based Practical Federated Learning on Blockchain. arXiv preprint arXiv:2304.05590 (2023).

Zihan Zheng, Peichen Xie, Xian Zhang, Shuo Chen, Yang Chen, Xiaobing Guo, Guangzhong Sun, Guangyu Sun, and Lidong Zhou. 2021. Agatha: Smart contract for DNN computation. arXiv preprint arXiv:2105.04919 (2021).