Optimistic Proof of Computation (OPoC): a compute efficient protocol for secure decentralized AI

Roku Sbaiashi

June 5, 2024

Abstract

In this paper, we introduce a novel consensus algorithm designed to facilitate secure computation within a decentralized computing framework. This algorithm ensures that for any given computation function $f()$ and input x, there are robust statistical guarantees that $f(x) = y$. Importantly, we integrate economic principles into the algorithm by leveraging token staking, which transforms these statistical assurances into tangible economic security. This economic security is crucial, as it underpins the integrity of interactions between AI models and smart contracts, particularly in environments where value is managed and the incentive to manipulate outcomes is significant due to the stakes involved. Optimized for resource-intensive computations, our algorithm is particularly adept at handling large language models and machine learning tasks, thereby enabling secure and reliable interactions between AI technologies and blockchain-based smart contracts.

1 Introduction

Running large language models (LLMs) and machine learning (ML) models, in general, is a demanding computational task. The development of a decentralized consensus mechanism that can ensure the correct execution of such computations is a complex challenge. It requires a delicate balance between latency, computational overhead, and correctness guarantees.

Traditionally, blockchain consensus methods require that all network participants verify whether state changes—such as balance adjustments or unspent transaction outputs—comply with the protocol-defined rules, and whether the block proposer has either performed significant computational work (Proof of Work, PoW)[Nak08] or holds a substantial stake in the network (Proof of Stake, PoS)[KN12] [BG17]. While theoretically possible, applying traditional consensus logic to the computation of LLMs and ML models would be practically infeasible due to the significant computational and data availability costs involved, which would scale linearly with the number of nodes participating in the consensus.

A possible approach to solve such a problem is to generate an easily verifiable proof that the computation has been executed correctly. Just like in proof-of-work all of the network participants can easily verify that the SHA-256 hash of a nonce combined with all the transactions included in a block - treated as a 256-bit integer - is less than a dynamically adjusted target, in our case, a similar verification process could be achieved by using a Zero Knowledge Proof system. Network participants could easily verify that an output y is the direct result of the execution of a specific AI model $f()$ on an input x without having to re-run the whole computation $f(x) = y$ but only by validating a ZK-SNARK proof. Although theoretically feasible, this approach currently faces practical limitations that prevent its application in scenarios requiring intense computation, such as with LLMs, or where latency constraints demand responses within seconds. Such a limitation is well understood in the cryptocurrency space, in a recent article [But24], Vitalik Buterin has categorized it as "Cryptographic overhead". Currently, generating a ZK-SNARK proof for even a modestly sized LLM, with just 1 million parameters, takes approximately 1000 seconds [SCJ+24][ezk23]. This is significant, especially considering that lower specification models in contemporary usage typically feature at least 7 billion parameters. Furthermore, the computational cost to generate such a proof is two to three orders of magnitude greater than the computation being verified, with a memory footprint potentially reaching terabytes [Lab23] [SCJ+24] [Lab24].

2 Optimistic Proof of Computation (OPoC) consensus

2.1 Architecture

To address the practical bottlenecks related to global computation footprint and latency, and to facilitate a wide range of secure computation use cases, including the interaction between AI models and smart contracts, we propose a novel consensus algorithm termed Optimistic Proof of Computation (OPoC). OPoC alleviates the need for all nodes to perform computations for each inference. Instead, network-wide computation and verification are only triggered if a disagreement occurs among a limited subset of validators v. These validators are randomly selected [MRV99] from the entire pool of validators V to perform the inference. The OPoC consensus process is divided into two stages: The first stage provides probabilistic assurance of the computation's correctness. Should any disagreements arise in this initial phase, the second stage offers a resolution mechanism to address such discrepancies. By convention, we'll call a validator honest if he/she follows the protocol and byzantine otherwise. We typically assume strictly less than p = 1/3 of validators are byzantine. This constant can be traced to Practical Byzantine Fault Tolerance (PBFT) from [CL+99], a classic consensus protocol in byzantine tolerance literature. In OPoC an actor requiring an inference from the network receives a probabilistic guarantee of correct computation. This guarantee is equivalent to the probability that at least one validator in the subset v is honest $v(honest) \geq 1$. Conversely, for an attacker to successfully propagate a malicious inference across the network, it would require that none of the validators in the randomly chosen subset v are honest. If only one of the v validators is honest, there would be a disagreement with the rest of the byzantine validators in v, and the consensus would scale to a larger subset of validators in V. Under the assumption of an honest majority, this expanded subset of V can determine the correct inference. Each validator participating in the network is required to stake an s amount of tokens with economic value; the tokens of the byzantine validators are subjected to slashing.

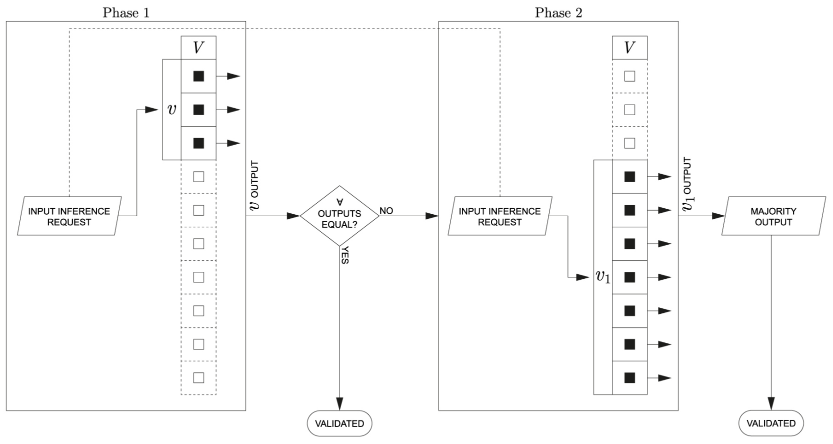

Figure 1: OPoC high-level architecture

2.2 Probabilistic guarantees on computation correctness

We can leverage the hypergeometric distribution formula to formally define the probability that a malicious inference successfully propagates through such an OPoC-enabled network and, thus, the probability of an attacker succeeding without being slashed.

The probability mass function (PMF) for selecting h honest validators out of v drawn from a total of V validators, where there are H honest validators, is given by:

$$P(X = h) = \frac{\binom{H}{h} \times \binom{V-H}{v-h}}{\binom{V}{v}}$$

Where:

- $P(X = h)$ is the probability of selecting exactly h honest validators from the subset of v participating validators.
- $\binom{H}{h}$ calculates how many ways we can select h honest validators from the total H honest validators.
- $\binom{V-H}{v-h}$ calculates how many ways we can fill the remaining slots in our subset with Byzantine (dishonest) validators, given that we've already selected v honest ones.
- $\binom{V}{v}$ is the total number of ways to select any v validators out of the entire pool of V validators, without concern for whether they're honest or Byzantine.

In this context, consider a scenario where the consensus algorithm operates under the assumption that 2/3 of the validators V are honest. If the total number of validators V is 100, and a subset of these, v, totaling 10 validators, are selected to vote on a computation, the probability of selecting a subset with zero honest validators (given $h = 0$ honest validators in this subset) would yield a probability of 0.0000076 .

3 Economic Security

To determine the economic security of each inference produced by the OPoC consensus network, we need to calculate the minimum reward that a rational Byzantine validator would accept to counterbalance the risk of having its stake s slashed. Assuming no collusion among single defecting validators, the Minimum Defecting Reward can be calculated as follows:

$$\text{Reward}(\text{defect}) = \frac{s}{P(X = h)}$$

Where $h = 0$

As long as the result of an inference generated by the network is directly or indirectly connected to a value smaller than the defined minimal reward to defect, there is no rational incentive for a validator to attack the network. Consider a scenario where the stake s is \$ 10,000, and the consensus algorithm operates under the assumption that 2/3 of the validators V are honest. If the total number of validators V is 100, and a subset of these, v, totaling 10 validators, are selected to vote on a computation, the economic security for each computation would be approximately \$ 1,315,789,473.

3.1 Maximally adversarial environments

To account for a maximally adversarial environment, we reconsider the no collusion assumption among single defecting validators. We assume a scenario where a single entity controls all of the 1/3 dishonest validators. In such a scenario, to prevent dishonest validators from avoiding participation in validation rounds where there is at least one honest validator ($h \geq 1$), we should modify the first stage of the OPoC consensus into two consecutive stages. In T_0, one validator performs the computation, while the rest of the validators in v perform it in T_1. Validators are chosen via a random function, ensuring that the validator selected in T_0 cannot predict which validators will be selected in T_1. To accommodate this new validation mechanism, we modify the previously defined PMF as follows:

$$P(X = h) = \frac{\binom{H}{h} \times \binom{(V-1)-H}{(v-1)-h}}{\binom{(V-1)}{(v-1)}}$$

This equation accounts for the subtraction of one dishonest validator from the total set V and the subset v. Applying the simulation data used for the no collusion scenario to this new formula for the maximally adversarial environment results in a reduced economic security for each computation to approximately \$ 450,450,450.

This approach allows us to provide flexible probabilistic assurances of the correctness of a required inference, limiting the computational footprint to a fraction of that required by complete validation by all network participants. An actor requiring an inference can set the probabilistic assurance based on the specific use case or value at stake.

Note on Inactivity Leak:

The Minimum Defecting Reward calculation for the maximally adversarial environment does not account for the economic cost applied through partial slashing to the defecting validators during the validation rounds they avoid participating in. The Inactivity Leak section describes such an additional disincentive to misbehave.

4 OPoC Security and Parallel Inference Scaling

4.1 Security

In terms of scalability, OPoC presents a significant structural advantage compared to traditional PoS and PoW consensus mechanisms. In both PoS and PoW, the total computational effort required to verify transactions increases linearly with the number of nodes joining the network. Conversely, OPoC operates differently: as the number of validating nodes V grows, the algorithm maintains the same level of statistical assurance on the correctness of the computations without a corresponding increase in the total computational effort used. This is because the ratio of participating validators v/V—the proportion of total validators V engaged for each inference—necessary to achieve consistent probabilistic assurances on the correctness of an inference scales sublinearly with the growth of V.

To elaborate, for a fixed ratio of v/V and a constant proportion of honest validators H/V, the increase in the overall population V results in an exponential decrease in the probability that v consists solely of Byzantine validators. This demonstrates the efficiency of OPoC in leveraging larger validator populations to enhance security without proportional increases in computational demand.

The following are the probabilities of encountering all dishonest validators among the randomly selected voting validators v, assuming $v/V = 3\%$ and $H/V = 2/3$, calculated across different network sizes $V = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$:

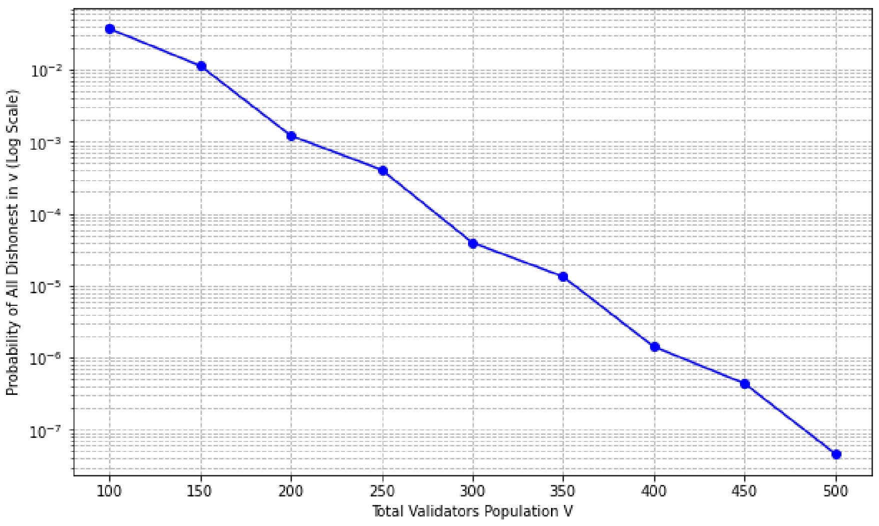

Figure 2: Probability of all byzantine validators in v

As a consequence of this exponential decrease in the probability of having only Byzantine validators within v, coupled with the linear growth of the population V, we can infer that the percentage of participating validators v/V necessary to maintain the same level of inference security decreases polynomially with the increase in the total population of validators V. To substantiate this assertion, we consider the hypergeometric distribution probability mass function (PMF) for the scenario of selecting exactly zero honest validators within v:

$$P(\text{all dishonest}) = \frac{\binom{V\cdot(1-h)}{v}}{\binom{V}{v}}$$

To determine how many validators v need to participate in order to achieve a certain P_{target} — the probability of selecting all dishonest validators — with changes only in the total number of validators V, we solve the following inequality for v:

$$\frac{\binom{V\cdot(1-h)}{v}}{\binom{V}{v}} < P_{\text{target}}$$

This problem can be tackled iteratively by simulating different levels of validator participation (v/V) and varying the total count of validators V.

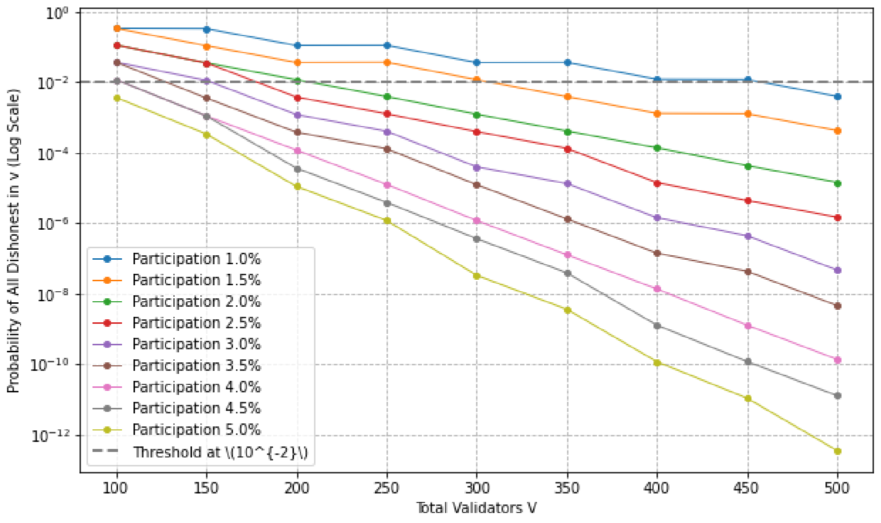

Figure 3: v/V validators needed for a $P_{\text{target}} < 10^{-2}$ of selecting all dishonest validators in v

The chart demonstrates that with a specific target for $P(\text{all dishonest})$ — in this case, $P_{\text{target}} < 10^{-2}$ — as V increases, v/V can be reduced. This illustrates how the OPoC consensus algorithm scales parallel computation capability linearly with the number of participating nodes while simultaneously enhancing network security and decreasing the necessary proportion of validators required to achieve a given level of security.

4.2 Parallel Inference

Under traditional consensus paradigms such as PoS and PoW, all validators are required to perform the same computations to validate transactions. This restricts the network's ability to run parallel inferences and thus limits the capability to serve multiple agents simultaneously. Furthermore, adding more validators to such networks does not increase the capacity for producing secure inferences per unit of time; instead, it merely amplifies the computational overhead required per secure inference.

In contrast, with the OPoC model, as demonstrated in previous sections, for fixed level of security assurances on the computation, at the growth of the total population of validators V we can maintain a fixed number of randomly extracted participating validators v for each inference. This means that under the assumptions that a) each request fully saturates the computational capability of each validating node - pessimistic assumption - and b) there is no disagreement between v validators; the capability of the computation network to perform parallel inferences scales linearly with the growth of the total validators population V and is equal to V/v.

$$\text{Number of parallel computations} = \frac{V}{v}$$

This scaling property is crucial not only for the computational efficiency of the network but also in terms of the economic sustainability of the consensus algorithm. Specifically, in a computational network powered by OPoC, adding nodes—which represent costly resources—directly translates into increased throughput. This, in turn, enhances the network's capacity to generate revenue. Therefore, OPoC not only optimizes computational resources but also aligns economic incentives with network growth and performance.

5 Resolving preliminary consensus disagreement:

5.1 Preliminary Consensus Phase

In the initial phase of the OPoC consensus mechanism, a subset v of the total validator population V executes the requested computation. If consensus fails during this phase—indicated by at least one validator in v disagreeing on the output y of the function $f(x)$—a method is required to resolve the disagreement and determine the correct result of y.

5.2 Brute Force Resolution Approach

A straightforward approach to reach consensus on the correct y involves scaling the subset of nodes v to a size v_1 that guarantees a majority of honest nodes, under the global honest majority assumption:

$$v_1 = V \times ((1 - H/V) \times 2) + 1$$

This brute force method is effective in resolving disagreements on the computation result y and is viable under the assumption that such consensus failures are rare and the s stake of byzantine proposers can be slashed. However, it is computationally expensive as it requires a significant majority of the network, v_1, to re-run the entire computation.

5.3 Efficient dispute resolution approach

To dramatically reduce this overhead, we implement resolution method inspired by Refereed Delegation of Computation (RDoC)[CRR13], a method proposed by Canetti, Riva, and Rothblum . RDoC allows a client to verify the correctness of a computation in the cloud when two servers, one of which is honest, disagree on the result. The client engages in a protocol with each server, using binary search to identify inconsistencies between intermediate states of their computations. When an inconsistency is found, the client can determine the cheater by verifying a single step of the computation. Collision-resistant hash functions enable servers to commit to the large intermediate states of the computation using compact commitments.

In our OPoC context, instead of a client and two servers, we have a network V verifying two different outputs from a computation $f(x)$ performed by a subset v. The dispute resolution process is not interactive but we require each participant in v to deliver both the final result and hashes of intermediate states at defined steps of the computation. These validators also temporarily store and, if a disagreement arises, broadcast the intermediate states to the entire network. Using collision-free hash functions, the extended subset v_1 can efficiently identify at which step the computation diverged and verify the correctness of the hashes for the inputs and outputs with minimal computational overhead. Upon detecting a disputed step, the involved nodes must broadcast the contentious input and output states for v_1 to verify the correct state transition.

5.3.1 Computational Cost Analysis

The computational cost of running a complete AI model computation, such as for a large language model (LLM), is given by:

$$C_{\text{full}} = N_{\text{layers}} \times C_{\text{layer}}$$

Where:

- C_{full} represents the total computational cost.
- N_{layers} indicates the number of layers in the model.
- C_{layer} denotes the cost to run a single layer, often determined by matrix operations and activations.

$$C_{\text{layer}} = 2 \times D_{\text{in}} \times D_{\text{out}} \times (N_{\text{params}} + N_{\text{activations}})$$

For partial computations, such as running a fraction of the model:

$$C_{\text{partial}} = k \times C_{\text{layer}}$$

Where k is the fraction of the total computation executed.

The efficiency of computation can be assessed by:

$$R_{\text{efficiency}} = \frac{C_{\text{full}}}{C_{\text{partial}}} = \frac{N_{\text{layers}}}{k}$$

In the context of OPoC, ignoring the computational cost for hash comparison that is negligible, the computational overhead of the verification of a disputed inference can be defined as follows:

$$C_{\text{verify}} = C_{\text{partial}} \times V \times ((1 - H/V) \times 2) + 1 = C_{\text{partial}} \times v_1$$

This technique allows for extremely reduced computational overhead for the verification of a disputed inference; in fact, theoretically, there is a linear relation between k and C_{verify}. On the practical level though, there is a trade-off between the dimension of k and the overhead for the first part of the OPoC consensus since the nodes participating in v need to ephemerally store the result of the intermediate states, an extremely small k would imply demanding memory requirement for the participating nodes in v. It is safe to assume that this technique can deliver at least two orders of magnitude in computational overhead reduction for a disputed inference verification compared to the "Brute force resolution approach" without impairing the efficiency of the first phase of the OPoC consensus algorithm.

6 Inactivity leak

The OPoC consensus algorithm provides a probabilistic guarantee on the correctness of a computation under the assumption of an honest majority among a population V of nodes, with a subset v of nodes participating in each inference. OPoC offers strong guarantees of having an active set of validators/verifiers compared to other consensus algorithms for efficient AI computation like opML[CSYW24], which are inherently affected by the Verifier Dilemma problem. The core of the Dilemma is that the incentive for a verifier to run the validation - operating expensive GPUs - is a not-guaranteed reward for eventually discovering a byzantine computation submitter. Fixing it requires complex incentive/disincentive mechanics. In OPoC, there is no ontological distinction between proposer and verifier of a computation. Each node is both a proposer and a verifier of computation as part of the validator population. This alignment removes any disparity in direct incentives between proposers and verifiers, as all validators are rewarded through token emissions and inference fees, ensuring an engaged set of participants.

Despite these strong incentives, there remains the challenge of inactive nodes within V, which could diminish the probabilistic assurances of computation correctness and disrupt computational finality. To address this, we introduce an "inactivity leak" slashing mechanism, inspired by strategies proposed in the CFFG paper [BG17] [EE22]. This approach aims to mitigate the risks associated with inactive validators. Under this mechanism, if selected validators fail to perform their assigned inference tasks, computational finality cannot be achieved. Consequently, no rewards are distributed across the network, and inactive validators are penalized. Slashing involves reducing the stake of inactive validators by an amount proportional to the rewards they would have earned during their period of inactivity and the duration of that period. This ensures that validators have a continuous disincentive to be inactive coupled with the incentive to be part of the validator set, thus supporting the network's overall reliability and the integrity of its computations.

7 Deterministic computation

Determinism, defined as the ability to produce consistent and repeatable results across different computing environments, is an essential enabler of the OPoC consensus framework. Particularly when handling machine learning operations that can suffer from inconsistencies due to inherent randomness and variability in floating-point arithmetic.

To counteract the randomness, it is standard practice to stabilize the outputs by setting a constant seed in the pseudo-random number generators. This step ensures that the "randomness" used in algorithms is repeatable and predictable. More complex issues arise with floating-point computations, especially given that different hardware platforms may not always yield identical results when processing the same floating-point operations. This discrepancy stems from rounding errors that occur in operations like floating-point addition, which can yield different results depending on the order of operation due to the non-associativity of floating-point arithmetic. To tackle the issues that arise from floating-point computations, OPoC uses quantization [JKC+18][Kri18], a transformative technique in machine learning that modifies continuous or high-precision numerical data into a more manageable, lower-precision format. This method solves the deterministic issues and is particularly valuable for optimizing the deployment of complex models on resource-constrained platforms. Quantization effectively reduces the broad spectrum of floating-point values to a finite set of discrete values, typically integers or low-precision floating-point numbers. This process can be described in four key steps:

1. Range Determination: Initially, the full range of the dataset, from minimum to maximum values, is identified. This range could represent the values of neural network parameters like weights or activation functions.

2. Scaling Factor Calculation: The identified range is scaled down using a scaling factor S, determined by the formula:
$$S = \frac{Range_{max} - Range_{min}}{2^n - 1}$$
where n represents the number of bits in the quantized data type.

3. Rounding: Once scaled, these values are then rounded to the nearest integer to fit the new, lower-precision format.

4. Reverse Mapping: Finally, the integers are converted back into the original data type using the inverse of the scaling factor, effectively mapping them to their new quantized values.

The quantization of a numerical value x can be mathematically expressed as:
$$Q(x) = round\left(\frac{x - Range_{min}}{S}\right)$$
where $Q(x)$ denotes the quantized representation of x.

It has to be noted that a trade-off exists between the floating-point reduction and the accuracy of DNN models, yet it is always possible to apply quantization to avoid rounding errors with a minimal impact on the overall model's accuracy reduction.

References

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[But24] Vitalik Buterin. The promise and challenges of crypto + ai applications, 2024.

[CL+99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[CRR13] Ran Canetti, Ben Riva, and Guy N Rothblum. Refereed delegation of computation. *Information and Computation*, 226:16–36, 2013.

[CSYW24] KD Conway, Cathie So, Xiaohang Yu, and Kartin Wong. opml: Optimistic machine learning on blockchain. *arXiv preprint arXiv:2401.17555*, 2024.

[EE22] Ben Edgington and Altair Edition. Upgrading ethereum, 2022.

[ezk23] ezkl. Honey i snarked the gpt, 2023.

[JKC$^+$18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19(1), 2012.

[Kri18] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[Lab23] Modulus Labs. The cost of intelligence, 2023.

[Lab24] Modulus Labs. Chapter 14: The world's 1st on-chain llm, 2024.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.

[Nak08] Satoshi Nakamoto. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*, 9:15, 2008.

[SCJ$^+$24] Tobin South, Alexander Camuto, Shrey Jain, Shayla Nguyen, Robert Mahari, Christian Paquin, Jason Morton, and Alex'Sandy' Pentland. Verifiable evaluations of machine learning models using zksnarks. *arXiv preprint arXiv:2402.02675*, 2024.