# Decentralized AI: Meaningful utilization of computational power for real-world application

David Liberman
liberman@gmx.com

**Abstract.** This paper explores the development of a decentralized AI infrastructure designed to address the escalating computational demands of artificial intelligence. Despite the potential of decentralized systems, a significant challenge remains: much of the computational power in decentralized networks is currently dedicated to tasks that do not contribute to real-world utility. The aim is to construct a decentralized AI network that optimizes the use of hardware resources, aligning computational efforts with productive tasks. To achieve this, the "Transformer-based Proof-of-Work (PoW)" or "Sprint" mechanism is introduced as an advanced consensus protocol tailored for decentralized AI blockchain networks.

## 1. Introduction

Artificial Intelligence (AI) has become integral to various industries, driving innovation and transforming workflows across numerous domains. However, the increasing sophistication of AI systems has led to a dramatic rise in computational demands, particularly for key tasks such as training large models and executing inferences. Training involves optimizing model parameters by processing extensive datasets, requiring high-performance computing resources such as GPUs and TPUs. Inference, or applying trained models to new data, also demands significant computational power, particularly as models handle increasingly diverse and complex operations. These meaningful activities (AI model training and inference) are the primary focus of the proposed decentralized AI infrastructure, ensuring that computational resources are dedicated to productive outputs. Unlike many decentralized systems, where much of the computational power is misused on meaningless tasks, the novel decentralized AI network aims to handle these essential tasks efficiently, with minimal resource waste, making the infrastructure scalable and sustainable.

Many decentralized systems today suffer from a fundamental misalignment in resource allocation and incentives. A significant portion of rewards is directed toward network security rather than directly advancing meaningful AI computation, which wastes block subsidies. For example, in systems like Bittensor, 60% of rewards are allocated to staking, which, while necessary for network security, does not contribute to AI computation. Furthermore, the remaining 40% of rewards are also used inefficiently, as contributors to AI computation often repeat tasks to ensure accuracy and meet network validation requirements, leading to suboptimal hardware utilization. This misallocation of incentives reduces efficiency, diverting capital and computational power from meaningful AI advancements, thereby increasing costs.

The proposed decentralized AI network addresses these inefficiencies by ensuring that almost 100% of computational resources are directed toward meaningful tasks such as AI model training and inference. This is achieved through time-bound Transformer-based Proof-of-Work competition (Sprint), as well as honest-majority-based validation, both described in detail below. This alignment of incentives with productive work eliminates subsidy wastage and optimizes the use of available hardware, providing a more efficient and cost-effective alternative to traditional infrastructures.

## 2. Centralization, decentralization, and main challenges

### Centralization

The scalability and efficiency of AI infrastructure are crucial, as they directly impact AI systems' cost and performance. However, reliance on centralized infrastructure, primarily controlled by a few major cloud providers, introduces several critical challenges:

1. **High costs and monopolistic pricing.** Centralized cloud providers offer computational resources at a premium, structuring their pricing models to favor large-scale enterprises. Smaller developers, lacking the bargaining power and economies of scale, are subjected to significantly higher costs. [1] This disparity is particularly evident when deploying advanced open-source AI models, such as Meta's Llama 3.1 405B, on leading cloud platforms, such as AWS or Azure. The expenses of running these models are comparable to those of proprietary solutions like OpenAI's API. This parity in costs results from monopolistic pricing strategies employed by major providers. These providers maintain high margins for smaller clients while offering substantial discounts to larger entities [2], sometimes even operating at a loss to secure their business. As AI models continue to evolve, incorporating not just textual but also visual and auditory capabilities, the cost per user escalates significantly. Developers utilizing these advanced AI models face increasingly prohibitive expenses, particularly when relying on centralized cloud infrastructure. This is compounded by the fact that internet users, accustomed to free or low-cost services, are often unwilling to cover these rising costs, placing additional financial burdens on developers. Historically, the perceived affordability of open-source large language models (LLMs) was due to their smaller size. However, the associated costs have similarly increased as these models have expanded to match the parameter counts of proprietary models from OpenAI and Anthropic. Smaller cloud providers find it difficult to compete due to lower utilization rates, which prevents them from offering competitive pricing over the long term. This pricing dynamic forces many developers to consider purchasing hardware outright [3], which can reduce costs but sacrifice the flexibility needed to manage the variable demand generated by user requests effectively. Consequently, the current cost structures not only inhibit innovation but also restrict access to advanced AI capabilities, creating significant barriers for smaller entities in the AI ecosystem.
2. **Risks of censorship and centralized control.** Concentrating computational resources within a few dominant providers presents significant risks related to censorship and centralized control. These centralized entities have the authority to impose restrictions, monitor usage, and potentially censor applications that do not align with their policies. As AI systems continue to develop and disrupt various industries, this concentration of power becomes increasingly problematic. Instead of driving down costs and democratizing access to AI technologies across all sectors, the control exercised by a small number of players could lead to a scenario where the economic benefits of increased efficiency are captured by these few entities rather than being distributed more broadly. Such centralization threatens the openness and accessibility of AI technologies, stifling innovation by limiting the diversity of applications and models that can be developed and deployed. The potential for censorship and restricted access underlines the need for a decentralized AI infrastructure that not only addresses the issues of cost but also ensures that the benefits of AI are available to all without the risks associated with centralized control and censorship. By decentralizing AI resources, the technology can be made more equitable and accessible, fostering a future where AI provides widespread benefits and mitigates the risks of monopolistic control and undue influence.

### Decentralization

Decentralization is often seen as a solution to the challenges of centralization, offering a way to distribute computational resources across a network of nodes of relatively small individual computational power, thereby eliminating the monopolistic control of centralized providers.

Modern decentralized networks also utilize additional incentive mechanisms, such as issuing new digital currency and providing a reward (known as block subsidy) with each transaction. These subsidies can

effectively lower the cost of hardware and computational resources for the customer, especially during the initial stages of network development. In the context of AI, the subsidies have the potential to reduce the cost of training and inference for developers and customers by as much as 50% or more compared to current centralized providers, particularly during the network's initial growth phase.

The recent AI advancement brought up a new computational challenge that forced large and small players to invest billions in hardware. However, computational costs are expected to decrease substantially over time, both following Moore's law and because of the proven potential of crypto-subsidized networks to supercharge competition and breakthroughs in computational technology. As these costs decline, the need for continued subsidies diminishes, making the incentive structures of decentralized systems well-suited for supporting AI's evolving computational requirements.

In theory, decentralized systems should provide greater cost-efficiency, flexibility, and resilience. However, the architecture of modern decentralized systems exhibits significant inefficiencies. A large portion of the block subsidies and computational power is consumed on tasks that do not yield meaningful results, primarily focusing on network security rather than practical outputs. This inefficiency stems from two main factors: the high cost of achieving consensus and the high degree of redundancy built into a computation accuracy verification.

Substantial computational or capital resources are required to secure the blockchain through mechanisms like Proof-of-Work or Proof-of-Stake. These consensus models assume that most participants are honest but require all participants to allocate significant resources to prevent malicious minority actors from manipulating the network by making everyone believe they represent the majority. *The reasons for inefficiencies are described in more detail in Chapter 3.*

In addition to securing blockchain, in decentralized AI projects, the same computational tasks are executed multiple times to ensure the accuracy of the results. Similarly, decentralized storage projects may duplicate files multiple times for verification. This redundancy leads to considerable resource waste, particularly in AI, where even small task repetitions can drastically increase computational costs. Moreover, even when results are verified by other nodes once or twice, there is no guarantee that these nodes are independent, and they may all be part of the same colluding group, thereby compounding the risk of false outcomes while maintaining the appearance of redundancy.

These inefficiencies also increase the environmental footprint, significantly raising energy consumption. The U.S. Energy Information Agency  estimates that cryptocurrency mining accounts for 0.6 to 2.3 [4] percent of global electricity use annually, highlighting the high energy consumption associated with these processes. While meaningful AI tasks also require significant electricity, the inefficiencies caused by task repetition and consensus mechanisms result in a far greater waste of resources in current decentralized systems.

**The need for decentralized AI infrastructure**

Despite these challenges, there is a compelling need for decentralized AI infrastructure to overcome the limitations of centralized systems. Decentralized networks can foster a genuinely competitive market environment, unlike the oligopolistic nature of centralized cloud providers. In a decentralized ecosystem, numerous participants can contribute computational resources and benefit from the network's growth, promoting innovation and efficiency. Furthermore, issuing tokens or coins can enable significant upfront subsidies for hardware and software.

## 3. Traditional consensus mechanisms and their inefficiencies

**Proof-of-Work**

Bitcoin's security isn't only based on complex cryptographic algorithms. It fundamentally relies on the principle of honesty among its participants. Honest node, a computer running software, strictly follows predefined consensus rules, ensuring consistent and predictable behavior within the network. The Bitcoin white paper underscores the importance of this by mentioning "node" 38 times and "honest" 16 times, often together, to ephasize the trustworthiness of those operating the network. On the flip side, "attacker" and "attacker nodes" refer to those with malicious intentions. To achieve their goals, attackers modify the rules within their local copy of the software, attempting to undermine the integrity of the network. So, while cryptography and incentives help make attacks more complicated, the expectation that the majority of nodes will act honestly keeps Bitcoin secure. As the Bitcoin white paper states, "The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote." [5] The network's integrity is upheld through the honest participation of nodes, which collectively control more computational power than any cooperating attackers. But for this, they must continuously occupy a significant amount of computational hardware and electricity, constantly increasing their usage and raising the capital spent on keeping the system secure. This results in inefficiency where massive resources are allocated not toward generating value (in Bitcoin, 0% of incentives are directed towards participants performing productive tasks, as the entirety of rewards is consumed by the validation of non-productive computations) but rather maintaining a dominance of computational power, making security dependent on an ever-growing consumption of energy and hardware, with diminishing returns. In reality, the only thing Proof-of-Work is trying to achieve is to ensure that the honest majority has more votes in determining what is true.

**Proof-of-Stake**

The blockchain community has explored Proof-of-Stake as an alternative to Proof-of-Work, primarily to mitigate the cost of computing and the environmental impact associated with Proof-of-Work consensus mechanism. In Proof-of-Stake systems, participants lock up a certain amount of cryptocurrency in a frozen account, with voting power and validation responsibilities distributed based on the amount staked.

While Proof-of-Stake aims to maintain network security with lower energy use, it introduces significant capital costs associated with staking. In Proof-of-Stake systems, incentives are primarily directed toward capital holders rather than contributors to computational tasks, resulting in high costs for the network. Modern cryptocurrency systems often compensate this capital at rates significantly above market levels. For example, Bittensor currently offers an IRR of 18% (excluding the increase in coin market prices). In Bittensor, 60% of additional incentives go to those who do not contribute to computational work, clearly illustrating the inefficiency of this model. If Ethereum had been a Proof-of-Stake network from the start, nearly all of its $300 billion market capitalization (aside from a percentage allocated to ICO) — would have been paid as subsidies to capital rather than being invested in hardware subsidies. This capital could have been used to significantly advance AI by subsidizing substantial computational infrastructure.

## 4. Proof-of-Work reimagined: a new approach to efficient utilization of computational power

While Proof-of-Stake offers an alternative to Proof-of-Work by aiming to reduce computational cost and energy consumption, it introduces challenges related to resource allocation and network efficiency. This has led to reconsidering what defines an effective and equitable consensus mechanism in decentralized networks. In crypto projects, truth is established through protocols designed to ensure that the majority genuinely represents the majority and is not merely an illusion created by a deceptive minority. This raises a fundamental question: How could the majority be determined effectively?

The proposed novel consensus algorithm combines both key elements of Proof-of-Work and Proof-of-Stake to create a balanced and efficient network security and governance mechanism *(see Appendix A. Consensus comparison table)*. It incorporates the key element of Proof-of-Stake, where only one chain and a block are considered valid when the majority of participants sign it off. However, unlike the traditional Proof-of-Stake mechanism, where voting weight is based on the amount of cryptocurrency staked, in the proposed system, voting weight is determined by the amount of computational evidence generated during a competitive Sprint, similar to the Proof-of-Work process. Sprint is a structured, time-bound competition where hardware providers (from here — Hosts) utilize their computational resources to solve complex tasks within a defined timeframe. The outcomes of these tasks serve as evidence, determining the voting weight each Host holds in the network.

This new approach returns to the roots of Proof-of-Work, focusing on computational power but optimizing its use. It achieves efficiency by recognizing that the primary purpose of computational tasks (ensuring network security and fairness) can be accomplished within a limited timeframe. During this limited timeframe, Hosts fully utilize their computational power to generate evidence, which is then used to determine voting weight. This method frees up computational resources beyond Sprint timeframe, so that they can be directed toward meaningful tasks, enhancing the efficiency and sustainability of decentralized AI networks. *For a technical specification of the Proof-of-Work mechanism designed for this framework, refer to Appendix B. Transformer-Based Proof-of-Work: Aligning computational power with AI workloads*

During Sprint, all Hosts engage in a computationally demanding task designed to be resource-intensive but easy to verify. Similar to Bitcoin's Proof-of-Work nature, where finding the correct nonce to produce a hash with one leading zero requires about 16 attempts, achieving a hash with ten leading zeroes could require trillions of attempts. Once the correct nonce is found, any Host can easily verify its correctness.

Following Sprint, Hosts (holding voting weights from the previous Sprint or from the genesis block in the case of the first Sprint) record new weights based on the result of the recent Sprint. These weights are then utilized to evaluate whether consensus is reached for a specific block candidate. Instead of basing voting weight on the amount of staked cryptocurrency - as in Proof-of-Stake the system allocates voting weight based on the quantity of valid computational work each Host performed during Sprint, employing a "one-computational-power-one-vote" principle outlined in the Bitcoin whitepaper. The proportion of voting weight in the proposed system is equivalent to what is observed in standard Proof-of-Work systems, where voting weight is directly tied to the computational effort exerted. Unlike traditional Proof-of-Work systems, where computing wastes 100% of the time finding hash functions, the proposed method utilizes computational power for meaningful tasks, such as AI model training and inference. The allocated voting weights remain effective until the end of the cycle, after which a new Sprint determines the next set of weights.

Unlike other decentralized computational networks, in the proposed protocol, the resulting voting weights are not only used to maintain ledger records but also to distribute meaningful tasks between computational nodes and validate the results returned by these nodes. This approach significantly reduces the need for redundant computations. This is possible because, unlike in PoS, voting weights correlate with the actual computational capacity of the nodes, and unlike in PoW, computational capacity is not occupied by meaningless tasks. The details of this process are described in *Appendix B. Transformer-Based Proof-of-Work: Aligning computational power with AI workloads.*

This method of allocating voting weights directly influences how rewards are distributed within the network. Rewards are strictly allocated to Hosts who contribute to meaningful tasks, such as AI model training and inference. This ensures that Hosts are incentivized not just to secure the network but to contribute to its growth and functionality actively.

**Sprint**

Sprint is a critical component of the novel consensus mechanism. It begins simultaneously for all Hosts, ensuring fairness by eliminating any advantage based on timing. This synchronous start is akin to a "starting

gun" in a sprint, where no one is allowed to start earlier than the others. To ensure this, a random number that cannot be predicted or influenced in advance is generated by all Hosts with voting power. This randomness is essential for preserving the integrity of the process, preventing any minority group of Hosts from pre-computing or gaining an unfair advantage. *Appendix C. Random number generation outlines the process for generating this random number, which is vital for maintaining fairness and security in the decentralized AI network.*

The duration of Sprint is deliberately confined to a short, defined timeframe, typically around 10 minutes, ensuring that the computational effort is both concentrated and efficient. Sprint occurs at regular, precisely specified intervals on a cycle basis, synchronized with the creation of blockchain blocks (e.g., *every k*n-block* in the blockchain). This regularity integrates Sprint seamlessly into the network's operational rhythm. The parameter *every $k \times n$ blocks* is system-defined and subject to tuning. In the final implementation, this could correspond to a real-world duration like 24 hours. This interval must strike a careful balance:

- it should be long enough so that the Sprint period itself remains a small fraction of the total cycle, minimizing the overall computational load.
- it should not be so long that it becomes prohibitively expensive or risky to exit.

Sprint itself, it must be short and efficient, yet long enough to collect statistically meaningful and verifiable data. The final system parameters, including both the Sprint duration and the cycle interval, should be empirically determined through extensive testing to ensure optimal trade-offs between security, performance, and cost.

**The Task**

The computational power loading function is specifically designed to favor Hosts with hardware optimized for tasks similar to those required for training LLMs. To achieve this, the computational task executed during Sprint closely mirrors the structure of transformer calculations, ensuring that the Hosts who succeed are those equipped with the appropriate hardware for such advanced computations. *For a detailed explanation of the Proof-of-Work mechanism designed for this framework, which aligns computational power with AI workloads, please refer to Appendix B. Transformer-Based Proof-of-Work: Aligning computational power with AI workloads.*.

Sprint must not interfere with normal user requests. When user requests are present, they must be processed without delay. In the absence of user requests, to ensure full computational power utilization, the system executes an alternative function, or "dummy tasks," whose outcomes cannot be predetermined. This is analogous to Bitcoin's hash function calculations, which serve no purpose beyond securing the network.

## 5. Real-world utility

Using the proposed protocol, a decentralized network can emerge, where Hosts (hardware providers) contribute computational resources to the network. Their hardware will run software based on the described protocol ("nodes"). According to the protocol as described above, each Host will get assigned a voting weight corresponding to its actual computational power. Their nodes will continuously reach consensus and maintain a ledger of records similar to how it's done in networks like Bitcoin or Ethereum. At the same time, their computational power remains available for real-world utility tasks such as AI inference or training.

Hosts will be compensated (either by Developers or through the issuance of new cryptocurrency) for executing tasks, covering their costs, and generating profits to sustain their operations. The distribution of computational tasks is proportional to the processing power of each participating node, and the rewards are allocated accordingly. This ensures that Hosts receive a fair level of utilization and, consequently, a fair share of the rewards. *For an example of the reward system for AI inference tasks, see the Chapter "Incentives".*

However, in decentralized networks, we cannot assume all nodes are honest nor that they won't attempt fraudulent behavior. The following types of fraud are possible when nodes perform tasks:

- A node may fail to complete the assigned task, hoping to receive a share of the reward without contributing. The node could either fail to return a result or return a result inconsistent with what would have been generated by genuine work.
- A node might return a result using a simpler model (e.g., with fewer parameters), hoping to receive full compensation while reducing its actual costs.
- A node may attempt to insert misinformation, malicious manipulation, or advertising into the returned result.

To mitigate these risks, the system incorporates a set of countermeasures designed to ensure efficiency without causing excessive computational overhead.

# 6. Task verification

### Majority Verification

In decentralized networks, the results produced by individual nodes cannot be trusted without verification. When other nodes perform a verification, the assumption should be made that there is always a risk of collusion between the nodes that produce the result and the ones verifying it. To address this, the system relies on a principle of majority verification based on the voting weight of computational nodes (Hosts). Task verification is sufficient when Hosts representing more than 50% of the total voting weight confirm the result, as we rely on the assumption that the majority of Hosts, based on their voting weight, are honest. This approach provides a clear guideline on how many verifications are required and when the results can be 100% trusted.

This method reduces the number of redundant verifications compared to systems where every node must verify every task. However, even with this approach, multiple verifications are still required to maintain security and integrity, introducing a significant verification overhead. Randomized Task Verification is introduced to further optimize resource use, which complements majority verification by drastically lowering the number of necessary checks while maintaining trust in the network.

### Reward accumulation and distribution

Before delving into randomized task verification, it is important to understand how rewards are accumulated and distributed in the system. Since not every task can be verified instantly, rewards for completed tasks accumulate during the cycle between Sprints, awaiting full verification. The release of these rewards depends on whether Host has fulfilled three main criteria:

1. **Accuracy:** Host must not have been found guilty of producing false or malicious results.
2. **Work Proportion:** Host must complete its fair share of the total workload.
3. **Validation Proportion:** Host must fulfill its role in the validation process by verifying a proportional amount of work done by others.

These criteria ensure that Hosts are incentivized to complete both computational and validation tasks honestly and efficiently. If a Host fails to meet any of these requirements, its rewards may be reduced or forfeited entirely. This reward accumulation system protects the network from dishonest behavior by holding rewards in reserve until verification is complete, ensuring that rewards are only distributed for tasks that have been successfully and honestly validated.

With reward accumulation in place, the system is equipped to implement randomized task verification, a probabilistic approach designed to reduce the verification burden while maintaining security and integrity.

**Randomized task verification**

To further minimize verification costs without compromising trust, the system employs Randomized Task Verification. Rather than verifying every task, the network randomly selects a subset of tasks for verification. This process ensures that Hosts cannot predict which tasks will be checked, thereby discouraging malicious behavior.

Work produced by one Host is subject to verification by other Hosts, but this verification follows a random pattern akin to spot-checks. Just as parking violations are enforced through random checks, where compliance is encouraged by the risk of being penalized, Hosts in the network are subject to the same uncertainty regarding task verification. Malicious Hosts risk losing all accumulated rewards if caught, making it more attractive to comply with the network's requirements.

Each Host's voting weight determines the probability that they will be assigned a verification task. For example, a Host with 50% of the voting weight may be responsible for verifying 1 out of every 20 task results produced by a Host, while a Host with 10% of the voting weight may verify one out of every 100 task results. This system ensures that, collectively, Hosts verify approximately one out of every 10 tasks, with the majority of Hosts verifying one out of every 20 tasks within their cluster. Even if some verifiers are malicious, the honesty of the majority ensures that every task is eventually verified by a trustworthy Host, preserving the network's integrity. If a Host is caught producing false results, it forfeits all rewards earned during that cycle.

A probabilistic model ensures that any malicious Host will eventually be caught over a series of tasks. Since Hosts perform numerous tasks within each cycle, and rewards are distributed only at the end of the cycle, the likelihood of a malicious Host going undetected is minimized. Even if a few false results escape detection, the cumulative probability of catching malicious behavior over time makes cheating unprofitable.

Allocating votes based on computational power rather than capital ensures that Hosts contributing computational resources are effectively able to validate the work of others. This method reduces the resource waste typically associated with redundant checks in current decentralized utility systems, lowering the required repetition rate to as little as 1-10%.

The validation process is further strengthened by a pseudo-random algorithm that governs how Hosts select which tasks to validate. This system ensures that the decision to validate a particular task is not purely random but is instead influenced by deterministic factors, making it possible to audit the selection process. For instance, each transaction is associated with a unique ID. A Host applies its private key to sign this ID, generating a signature that serves as a seed for determining whether the Host should validate the specific task. Importantly, this signature and the validation result can be shared with the network. Other Hosts can independently verify that the signature is authentic and that it correctly served as the seed for the random function, which determined that the Host was supposed to validate that specific task. This level of transparency ensures that all Hosts adhere to their responsibilities, allowing the entire network to maintain trust in the validation process.

Combining majority verification and randomized task verification, the system balances trust and efficiency, ensuring that tasks are verified rigorously while minimizing resource waste. Rewards are only distributed to Hosts that have passed all verifications, further reinforcing the importance of honest participation in the network.

**Verification challenges**

The process of validating whether a Host is malicious when it comes to AI inference or training presents significant technological challenges, particularly due to the non-deterministic nature of some hardware operations. Hardware tends to produce varying results even when using the same input and randomization seed. As a result, determining whether a Host is acting maliciously cannot be treated as a purely deterministic task; it must be approached as a statistical problem. This recognition means that conclusions

about whether a Host is engaged in a malicious activity are based on probabilities rather than certainties, with considerations for both false positives and false negatives.

When a result is flagged as potentially incorrect, it cannot be immediately assumed to be malicious. Instead, other Hosts, which together represent the majority of voting weight, must re-verify the result. This re-verification process takes into account the potential variability in outcomes, recognizing that even honest Hosts may produce slightly different results due to hardware differences.

This statistical approach to verification is particularly important in reducing the likelihood of false positives. The final decision regarding a Host's honesty might involve accepting a small number of false positives, allowing Hosts to make a limited number of errors before facing penalties. In this way, punishment is reserved for Hosts that consistently exceed an acceptable error threshold rather than being triggered by a single mistake.

*A potential implementation for the verification process is described in more detail in Appendix B. Transformer-Based Proof-of-Work: Aligning computational power with AI workloads..*

**Reputation**

Reputation can accumulate over time as Hosts continue to act honestly. New Hosts start with a reputation score of zero. As a Host successfully contributes to the network without being caught producing false results, their reputation increases with each subsequent cycle. The longer a Host remains honest, the higher their reputation grows.

A low reputation score increases the likelihood that a Host's work will be scrutinized — up to the point where every single task they produce may be verified. Conversely, Hosts with a higher reputation face less frequent checks. The reward a Host receives thus depends on how much of their work is verified by others. For example, if every 10th task is verified, a Host with a high reputation receives 100% of the rewards for nine tasks and only 50% for the one that is verified. If every task is verified, the reward is halved. If a Host is caught producing malicious work, they lose all rewards for that period, and their reputation is reset to zero, placing them under the highest level of scrutiny again.

*For further details, refer to Appendix D. Reputation and validation mechanisms.*

**Dummy tasks**

During periods of low user activity, the system may introduce mechanisms to allow Hosts to build a reputation by performing tasks that are not directly generated by real users. These "dummy tasks" are designed in two ways:

1. **Distinguishable Dummy Tasks:** Hosts differentiate real user requests and dummy tasks, ensuring that they allocate resources appropriately.
2. **Indistinguishable Dummy Tasks:** Alternatively, dummy tasks can be designed to be indistinguishable from real user requests, making it impossible for Hosts to prioritize or neglect tasks based on their origin. These prompts must be generated by an algorithm that closely mimics real user behavior, preventing Hosts from discerning whether the task is genuine or synthetic. Indistinguishable Dummy Tasks, designed to ensure that Hosts cannot prioritize or neglect tasks based on their origin, present unique challenges that must be addressed to maintain the integrity and efficiency of the network. These challenges are centered around ensuring task authenticity, anonymity, and fair distribution.

Below are the three primary issues that arise with the implementation of Indistinguishable dummy tasks:

1. **Indistinguishability of prompts nature:** The first challenge lies in ensuring that dummy tasks are indistinguishable from real user requests. This requires the algorithm that generates these prompts to be sophisticated enough that Hosts cannot differentiate between real and synthetic tasks. The generation process must be so robust that even with extensive analysis, Hosts cannot confidently

assert whether a user request is genuine or a dummy task. If Hosts were able to distinguish between these tasks, they might allocate resources differently, leading to an imbalance in the system and potentially undermining the fairness of the network.

2. **Unlinked transaction balances:** The second challenge involves the financial transactions associated with these tasks. All transactions, whether linked to real user requests or dummy tasks, must carry a balance that is not affiliated with the Host executing the task. This is to ensure that the incentive mechanism remains unbiased and anonymous. The protocol needs to account for the back compensation of funds to the entity generating dummy tasks. However, this compensation must be processed through random accounts rather than directly from the main account, preventing any association with specific Hosts. This level of financial decoupling is critical to maintaining the anonymity and fairness of task allocation.
3. **Anonymizing task origin:** The third challenge is related to the anonymity of the task's origin. To preserve user privacy and prevent Hosts from identifying the source of the request, the network is structured so that user requests are first routed through a random Host. This Host does not perform the task itself but instead determines which Host should complete it according to the network protocol. This mechanism ensures that the majority of user requests are anonymized, with the originating user's identity obscured. By randomizing the Host that initially receives the task and ensuring that this Host acts as an intermediary rather than a direct executor, the system effectively hides the user's IP address and further secures their anonymity.

**User-driven oversight**

The system also includes a mechanism for end-user-driven oversight. If a user suspects that a result is incorrect or suspicious, they can submit a paid report by clicking a "dislike" or "report" button. This action triggers a verification process by other Hosts. If the user's report is validated, the malicious Host forfeits its rewards, which are then shared with the user and the verifying Hosts. However, if the report is found to be incorrect, the Hosts involved in the verification process are rewarded instead. This mechanism ensures that the system heals itself by continuously reinforcing the network's integrity and trustworthiness through Validators (Hosts) and user oversight.

**Workload proof**

Finally, the system assumes that both workload and verification duties are evenly distributed among Hosts. It is penalized if a Host fails to complete its proportional share of tasks or verification duties. The specific penalties are determined by a formula that accounts for the Host's overall performance relative to the network's expectations. This ensures that every Host contributes fairly to the network's operation, maintaining the balance and integrity of the decentralized system.

**Bounty reward**

The rewards for identifying malicious Hosts must be carefully calibrated to ensure they do not exceed potential earnings from honest participation. This precaution prevents unintended incentives that might arise from discovering and penalizing malicious Hosts.

Moreover, if a Host is found to be engaging in malicious behavior when it processes a request paid by a user, the payment should be reimbursed, so it can't be used as part of the bounty reward. This consideration underscores the importance of maintaining user trust and ensuring the system remains equitable for all Hosts.

## 7. Inference

A network built on the principles described here can offer external Developers LLM inference services using widely adopted open-source LLM models, such as Llama 3.2 70B or Mistral Large.

Developers submit their users' requests to the network, which are then processed by the Hosts of Hosts using the specified model to generate responses. For Developers' convenience, these requests can be made through an API similar to those provided by centralized services like OpenAI Platform.

**Key participants in a decentralized AI ecosystem for LLM inference**

A decentralized AI infrastructure relies on the collaboration of several key participants:

- Hosts. Hosts contribute computational resources to the network and are rewarded based on the amount and quality of resources they provide.
- Developers. Developers build and deploy AI applications within the decentralized network, leveraging the distributed computational power to run their models. This approach provides Developers greater flexibility and reduced costs, as cloud providers' limitations and pricing models or closed-source LLMs do not constrain them.
- Users. End-users interact with AI applications running on the decentralized network. Their requests (inferences) generate the demand for computational resources, driving the growth and expansion of the ecosystem.

# 8. Training and fine-tuning

**Motivation and the Current State of Training**

Distributed machine learning faces multiple critical challenges that impede truly decentralized training at scale.

1. When Hosts join a distributed training network, they are fundamentally untrusted entities, with some potentially submitting fraudulent computations or malicious data. Any viable system must be able to identify these bad actors and implement appropriate penalties.
2. Unlike controlled data centers with high-bandwidth internal networks, decentralized training must operate over standard internet connections with their inherent bandwidth limitations and latency issues.
3. Traditional approaches to training large models require each Host to make gradient steps for the entire model locally—an unrealistic expectation given the demand for massive computational resources from every Host.
4. To date, most distributed training approaches have relied on centralized coordinator Hosts, introducing single points of failure and trust concerns.

Our approach aims to address these challenges through several solutions. We leverage DiLoCo's communication-efficient training approach [6] to significantly reduce bandwidth requirements between Hosts. We replace centralized coordination with on-chain management, eliminating single points of failure and enhancing system resilience. To address the issue of untrusted Hosts, we implement a proof-of-learning [7] inspired validation system that enables verification of training contributions. For scaling model size without requiring Hosts to store the entire model, we explore sharding approaches that allow Hosts to host only portions of the complete model.

*Each of these solutions is described in further detail in* *Appendix E. Decentralized Synchronization and Validation in Large-Scale Distributed Learning.*

To ensure long-term sustainability of these efforts, the network dedicates 20% of all inference revenue to support future model training. This guarantees that foundational AI development continues even beyond the early subsidized stages, aligning economic incentives with the evolution of the ecosystem. *See details in "Gonka: Tokenomics" document.*

### Fine-tuning

Fine-tuning provides Developers with the ability to fine-tune open-source models to meet specific tasks or applications. This process builds on the existing foundational models, allowing Developers to submit targeted data and request fine-tuning, similar to established practices in platforms like OpenAI. Participating Hosts in the network use the provided data to conduct additional training, forming supplementary layers that tailor models even further to meet Developers' needs, achieving improved results without the need for extensive prompts. These enhancements contribute to more efficient token usage and lower latency during inference. The decentralized nature of the network brings collaborative expertise into the fine-tuning process, fostering varied approaches that enhance model performance. This fine-tuning framework ensures that Developers can leverage enhanced, specialized versions of models for their unique purposes, maintaining flexibility while benefiting from the strengths of a decentralized infrastructure.

## 9. Incentives

### Dual-reward system

A dual-reward system analogous to the mechanisms observed in Bitcoin is proposed to incentivize participation within the network. Hosts who provide computational power are compensated through two primary avenues: transaction fees and a network-wide inflationary reward system.

Transaction fees are embedded within each transaction, similar to Bitcoin, where the end-user or Developer attaches a fee to execute specific tasks. This ensures that Hosts contributing computational resources are rewarded directly for their efforts.

In addition to transaction fees, Hosts receive rewards through an inflationary mechanism. Each block generated within the network issues a reward distributed proportionally among Hosts based on their computational contribution. This reward decreases over time, following a phased approach:

1. Initial subsidy phase: A substantial subsidy offsets the capital expenditure (CAPEX) associated with providing hardware in the network's early stages. This incentivizes early participation and ensures network growth. This subsidy will taper off as the network matures, transitioning the primary reward source to transaction fees, mirroring Bitcoin's diminishing block rewards.
2. Market efficiency: As subsidies decrease, market efficiency is expected to rise, driven by advancements in hardware performance. Just as the Bitcoin network saw the development of highly efficient ASICs, the network anticipates similar innovations, reducing the cost of computational power provision and ensuring sustainability as rewards shift entirely to transaction fees.

This dual-reward system also addresses the challenge of hardware utilization. In the network's launch phase, when hardware may be underutilized, initial subsidies distribute costs across fewer tasks, making CAPEX more manageable. As the network expands and hardware utilization increases, the need for subsidies diminishes, and transaction fees become the sole reward source. This progression allows the network to scale effectively while maintaining a competitive edge over centralized cloud providers, ultimately offering services at a fraction of the cost *(for detailed breakdowns of allocation strategy, subsidy schedules, and long-term sustainability models, see "Gonka: Tokenomics" document.)*.

### Transaction cost management

Transaction costs within the network are designed to be flexible and predictable, akin to Ethereum's GAS model or the OpenAI API's pricing structure. Users specify a maximum cost they are willing to pay, ensuring that tasks are performed within this predefined limit.

If computational resources are exhausted before task completion, the transaction is canceled, and the spent tokens are not refunded, similar to Ethereum's handling of exceeded GAS limits. The rewards generated from these transactions are distributed between the Host executing the task and the one verifying it, ensuring fair compensation across Hosts.

**Long-term sustainability**

The long-term strategy focuses on attracting Hosts willing to dedicate their hardware to the network, positioning the project as a cost-competitive alternative to traditional cloud providers. Leveraging decentralization, the network is designed to offer significantly lower prices for computational tasks, making it an attractive option for Developers and businesses seeking cost-effective solutions.

In conclusion, the tokenomics strategy is crafted to ensure the network's sustainability, support early adopters, and maintain competitive pricing. Balancing initial subsidies with a gradual transition to transaction fees establishes a robust ecosystem that rewards Hosts, drives hardware innovation, and positions the network as a leader in decentralized AI computation.

# 10. Conclusion

This paper has introduced a decentralized AI infrastructure utilizing the “Transformer-based Proof-of-Work (PoW)” consensus mechanism (“Sprint”), and it improves both centralized and current decentralized systems by addressing their core inefficiencies.

Current centralized AI infrastructures, dominated by a few major cloud providers, suffer from high operational costs, monopolistic pricing, and risks of censorship. These limitations restrict access to advanced AI technologies, especially for smaller Developers, while concentrating control in the hands of a few entities. In contrast, the proposed novel decentralized model democratizes access to AI resources, fostering innovation through competitive pricing. Unlike existing decentralized systems that rely heavily on inefficient consensus mechanisms such as traditional Proof-of-Work or Proof-of-Stake, where the majority of computational power is misdirected towards securing the network rather than performing valuable tasks, the proposed system ensures almost 100% of computational resources are used for meaningful AI tasks like training and inference. This approach eliminates resource waste in systems such as Bitcoin, where all incentives are consumed by network security, or Bittensor, where only 40% of rewards go towards AI computation. The proposed system maximizes hardware utilization by aligning computational efforts directly with productive outputs, significantly reducing the cost of operating AI models.

The novel Transformer-based Proof-of-Work (PoW) mechanism (“Sprint”) balances the strengths of both Proof-of-Work and Proof-of-Stake systems while eliminating their drawbacks. It adopts the fairness and security of Proof-of-Work, ensuring computational power is utilized effectively while avoiding the capital inefficiencies of Proof-of-Stake, where incentives disproportionately benefit capital holders rather than computational contributors. This hybrid model rewards Hosts solely for their contributions to AI workloads, ensuring that every unit of energy spent yields productive results.

The decentralized structure eliminates the monopolistic control associated with centralized cloud providers. Distributing computational tasks across a global network of Hosts offers greater resilience, transparency, and censorship resistance. Developers' ability to deploy AI applications on this decentralized infrastructure without the constraints of centralized pricing or policy control ensures more flexibility and autonomy, particularly for smaller Developers traditionally priced out of the market by centralized cloud giants.

The proposed decentralized AI system offers a better alternative to centralized and traditional decentralized systems. It reduces costs, increases resource efficiency, ensures meaningful use of computational power, and democratizes access to advanced AI capabilities.

## Table of contents

## Appendix A. Consensus comparison table

This table contrasts traditional consensus mechanisms—Proof-of-Stake and Proof-of-Work—with the proposed "Transformer-based Proof-of-Work (PoW)" mechanism ("Sprint"). It highlights how the new approach reallocates computational effort from security overhead to productive AI tasks, aligning incentives with real-world utility.

| **Aspect** | **Proof-of-Stake** | **Proof-of-Work** | **"Transformer-based Proof-of-Work (PoW)" mechanism ("Sprint")** |
|---|---|---|---|
| Voting Weight Basis | Based on the amount of cryptocurrency staked | Based on computational power and work performed | Based on computational work during Sprint |
| Task Focus | Mainly securing the network | Computational work dedicated solely to network security | Meaningful tasks such as AI model training and inference |
| Resource Allocation | Tied to the amount of staked capital | All resources are directed to computational tasks for network security | Resources allocated efficiently for both network security and productive tasks |
| Efficiency in Utilization | Efficient in terms of energy but less so in resource allocation | Inefficient, as 100% of computational work goes towards non-productive tasks | Optimized efficiency by directing resources toward useful tasks |
| Energy Consumption | Low energy consumption compared to PoW | High energy consumption | Reduced energy consumption by avoiding wasteful computational tasks |
| Reward Distribution | Mostly directed towards capital holders and securing the network | Entirely consumed by validating non-productive computations | ~100% allocated to productive, meaningful tasks |
| Incentives for Productive Work | Incentives are not strongly tied to computational contributions | No incentives for meaningful work. 0% are directed to productive tasks | Strongly incentivizes Hosts to contribute to AI advancements and productive tasks |

## Appendix B. Transformer-Based Proof-of-Work: Aligning computational power with AI workloads

The proposed Proof-of-Work mechanism is designed to leverage the computational characteristics of neural network architectures, effectively aligning the voting power with the actual workload of LLMs in a decentralized AI network. Hosts are incentivized to optimize their hardware and software for LLM computations.

The Proof-of-Work algorithm is structured to resemble the computational capacity for LLM inference and training. Every Host executes a parametrized function comprising computational blocks typical to LLM architecture illustrated in Figure 1. The computational capacity of a Host is demonstrated through the number of vectors found during Sprint, that satisfy a specific condition. A proof of computational capacity is sent to the network and validated by other Hosts. Voting power in the network is directly proportional to the computational capacity demonstrated during Sprint, ensuring fair representation based on actual contributed resources.

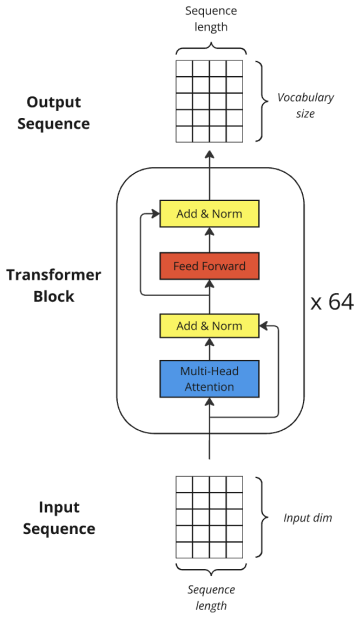



*Figure 1. Transformer architecture illustration*

Hosts are incentivized to optimize their hardware and software for LLM computations, benefiting both the consensus mechanism and the network's primary function of AI processing. The randomness of the generated model for each Sprint prevents pre-computation advantages.

### Sprint procedure

1. **Initialization**
   - **Sprint seed generation.** At the start of each Sprint cycle, a **Sprint Seed** is generated with a random number generator based on the latest blockchain state. This seed is used to initialize the shared parameters of the parametrized function. Sprint Seed is specific to each Sprint and the same across all the devices participating in that Sprint.
   - **Model initialization.** The parametrized function is a compact Transformer-based model (hereinafter referred to as Transformer) imitating LLMs on a smaller scale. This Transformer consists of 64 layers, each containing a multi-head attention mechanism with 128 heads and a feed-forward block. The model has an embedding dimension of 512 and uses a vocabulary size of 8192. The feed-forward block in each layer expands to a hidden dimension of 8192 before projecting back to

512. The model is designed to process sequences of length 4. Total number of parameters is about 2.3 billion.
    - **Node Seed creation.** After the model is initialized, every Host generates its unique Node Seed that is based on its public key.
2. **Iterations**
    - **Nonce iteration.** During Sprint, every Host iterates over nonce values (Input Seeds), which are combined with Node Seed and a current Sprint Seed produce input sequences to the Transformer. These sequences are passed to the Transformer to compute the output sequences, and only the last vectors of output sequences are used to construct a proof.
    - **Appropriate vectors.**
        - **i.** Host must find output vectors close enough to a **Target Vector** in terms of Euclidean distance. Vectors satisfying the described condition are called **Appropriate Vectors**.
        - **ii.** The **Target Vector** is randomly initialized at the beginning of each Sprint with a Sprint Seed and is common for all the Hosts participating in Sprint.
    - **Distance threshold**. The distance threshold is constant and is chosen such that the chance of finding an Appropriate Vector from one nonce is about 1 in 900 and can be used to control difficulty.
3. **Fraud prevention**
    - **Random permutation.** To prevent fraud, the output vector's coordinates are randomly permuted right before the calculation of the distance to the Target Vector. This procedure ensures that a Host can't abuse the continuity of the neural network, in particular, if a Host finds an Appropriate Vector, it could just iterate over nonce values to find an input vector close to the one that produced the Appropriate vector.
    - **Defining factors**. The nonce value, Node Seed, and Sprint Seed define this random permutation.
4. The more Appropriate Vectors are found, the more times the Transformer was inferred during a fixed time duration Sprint, which indicates higher computational capacity. This condition implements a similar principle to the Bitcoin Proof-of-Work. Finding an Appropriate Vector is analogous to finding the hash with a particular number of leading zeros.

**Validation of proof**

The proof of computational capacity is a set of nonce values. All of these nonce values should produce such input sequences that, if entered into the Transformer, they produce Appropriate Vectors.

The proof validation process consists of 3 steps:

1. Reconstruction of the Transformer and the Target Vector of Sprint with Sprint Seed, using the blockchain state at the time of Sprint.
2. Generation of the Host's input sequences using the provided nonce values and the Node Seed.
3. Performing Transformer forward passes, and confirming that all the provided nonce values result in Appropriate Vectors.

Voting weight is allocated proportionally to the number of valid proofs presented by the Host.

By implementing this LLM Proof-of-Work mechanism, the decentralized AI network achieves a harmonious balance between security, fairness, and practical utility, positioning itself at the forefront of efficient and purpose-driven blockchain consensus for AI applications.

**Code examples**

The examples below illustrate how these procedures might look in Python-like pseudocode, with some functions serving as conceptual placeholders rather than executable code. Sprint duration and distance thresholds are fixed constants.

Python

```python
def generate_proofs(node_public_key, latest_blockchain_state):
    """
    Generates valid nonces.
    For each nonce, the function generates input sequence and checks if
    this input sequence lead to an appropriate vector for the current Sprint.
    Every nonce that generate an appropriate vector is sent as a proof.
    Args:
        node_public_key (str): Node's public key.
        latest_blockchain_state (str): Current blockchain state.

    Returns:
        list : Valid nonces.
    """
    timer.start()
    Sprint_seed = generate_Sprint_seed(latest_blockchain_state)
    transformer_model = initialize_transformer(Sprint_seed)
    node_seed = generate_node_seed(node_public_key)
    target_vector = generate_target_vector(Sprint_seed)
    valid_nonces = []
    for nonce in nonce_generator:
        input_seed = combine_seeds(nonce, node_seed, Sprint_seed)
        input_sequence = generate_input_sequence(input_seed)
        output_sequence = transformer_model.forward(input_sequence)
        output_vector = extract_last_vector(output_sequence)
        permutation = generate_random_permutation(input_seed)
        permuted_vector = permute_vector(output_vector, permutation)
        distance = compute_euclidean_distance(permuted_vector, target_vector)
        if distance < threshold:
            send_valid_nonce(nonce, node_public_key)
            valid_nonces.append(nonce)
        if timer.check() > Sprint_duration:
            break
    return valid_nonces

def validate_proofs(node_public_key, submitted_nonces,
blockchain_state_at_Sprint):
    """
    Validates submitted nonces based on the blockchain state at Sprint time.
    Using a Sprint seed from the blockchain state and the node's public key,
    the function checks if each submitted nonce generates apprpriate vector.

    Args:
        node_public_key (str): Node's public key.
        submitted_nonces (list): Nonces to validate.
        blockchain_state_at_Sprint (str): Blockchain state at Sprint time.

    Returns:
        list: Validated nonces.
```

```
    """
    Sprint_seed = generate_Sprint_seed(blockchain_state_at_Sprint)
    transformer_model = initialize_transformer(Sprint_seed)
    node_seed = generate_node_seed(node_public_key)
    target_vector = generate_target_vector(Sprint_seed)
    valid_proofs = []
    for nonce in submitted_nonces:
        input_seed = combine_seeds(node_seed, Sprint_seed, nonce)
        input_sequence = generate_input_sequence(input_seed)
        output_sequence = transformer_model.forward(input_sequence)
        output_vector = extract_last_vector(output_sequence)
        permutation = generate_random_permutation(input_seed)
        permuted_vector = permute_vector(output_vector, permutation)
        distance = compute_euclidean_distance(permuted_vector, target_vector)
        if distance < threshold:
            valid_proofs.append(nonce)
    return valid_proofs
```

## Appendix C. Random number generation

A key element of Sprint is generating the random number that seeds the computational task. This random number must be truly random and immune to manipulation by any Host. Ensuring the integrity and fairness of this process is crucial, as it forms the basis for subsequent computations and consensus mechanisms in the network. To achieve this, the system employs a combination of classic cryptographic algorithms alongside advanced techniques like Threshold Cryptography with Synchronization and the Commit–Reveal with Timed Reveal mechanism. These methods are integrated to guarantee randomness, security, and decentralization at every step of the process.

- **Threshold Cryptography with Synchronization.** In the decentralized AI network, the random number is generated collaboratively by multiple Hosts using a Threshold Cryptography scheme. The secret (random number seed) is split into multiple fragments and distributed among a group of Hosts, each holding a share of the secret. To reconstruct the final random number, a predefined threshold of Hosts must combine their shares. This threshold system ensures that no individual Host influences the outcome or generates the number independently. Synchronization plays a critical role in ensuring that these Hosts cooperate in real time to reconstruct the number. By synchronizing their actions, Hosts prevent any delay, coordination failure, or manipulation of the process. This mechanism significantly strengthens the network's defense against malicious actors by distributing the responsibility for number generation and preventing any single point of failure.
- **Commit–Reveal with Timed Reveal.** In addition to Threshold Cryptography, the system incorporates a Commit–Reveal with Timed Reveal scheme to further secure the random number generation process. During the commit phase, each Host submits a cryptographically hashed version of its intended contribution to the random number. These commitments are locked and stored within the system, preventing any Host from altering its input after viewing the contributions of others. The reveal phase occurs after all commitments are received. At this stage, each Host reveals its original value, which is verified against the earlier commitment. The timed reveal aspect enforces a specific time window for the reveal phase, ensuring that all Hosts disclose their inputs simultaneously. This prevents any Host from withholding or manipulating their commitment, safeguarding the fairness and transparency of the process.

These methods ensure that the final random number is the product of contributions from multiple Hosts, each with voting weight proportional to their computational effort in the previous Sprint. By tying voting weight to prior contributions, the system maintains a balance of fairness and computational influence. This mechanism guarantees that no single Host can predict or influence the outcome, ensuring the random number remains unbiased and unpredictable.

At the end of each cycle, the group of Hosts with the highest accumulated voting weights must repeat this random number generation procedure just before creating the next k*n-block in the blockchain. The result of this block generation is a new random number, which will be used to seed the next Sprint. This continual regeneration of random numbers ensures that each Sprint is independent and unpredictable, further reinforcing the system's integrity and the equitable distribution of computational influence within the network.

# Appendix D. Reputation and validation mechanisms

Maintaining trust in a decentralized AI network requires mechanisms that incentivize honest behavior and minimize verification overhead. This appendix defines the reputation and validation system that dynamically adjusts scrutiny based on Host performance. By rewarding long-term reliability with reduced validation frequency and integrating probabilistic auditing, the network ensures efficient task verification, fair reward distribution, and user protection.

### Executor reputation system

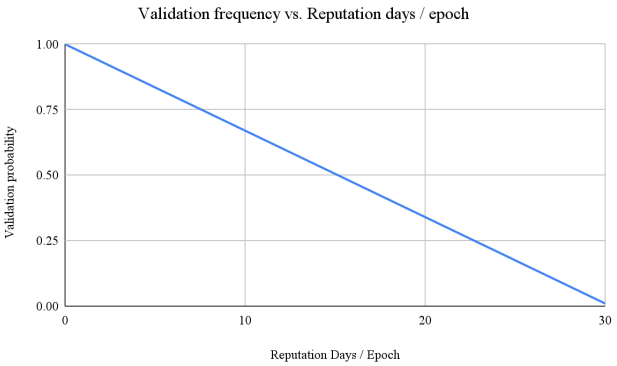
To ensure integrity and reliability, a reputation system for executors is implemented into the network, which dynamically adjusts validation frequency based on their historical performance. This system is designed to incentivize honest behavior and efficiently allocate validation resources.

### Reputation accrual

The longer an executor participates in the network without being detected for fraudulent activities (i.e., providing invalid results), the higher their reputation score becomes. The goal is to gradually reduce the validation frequency for high-reputation executors, transitioning from validating every task to validating only a small fraction of tasks. The reduction in validation frequency follows a linear progression:

$$\frac{1-(1-0.01)\times MIN(DAYS\ OF\ REPUTATION, N)}{N}$$

Initially, every task is validated (probability = 1.0). Over time, the validation probability decreases linearly until it reaches a minimum threshold (e.g., 0.01). The parameter N (default: 30 days/epoch) controls the duration of this transition. Increasing N extends the period before an executor reaches the minimum validation frequency.




### Dynamic validation frequency adjustment

The minimum validation frequency (e.g., 0.01) is contingent on the network's request volume. To optimize validation efforts, the system adjusts the validation frequency based on the number of requests processed per day/epoch:

- High volume (≥ 10,000 requests/day/epoch): The minimum validation frequency (e.g., 0.01) applies.
- Medium volume (100 < requests < 10,000/day/epoch): The validation frequency adjusts linearly. For example, at 5,000 requests/day/epoch, the minimum validation frequency is 0.05.
- Low volume (≤ 100 requests/day/epoch): Every task is validated (validation frequency = 1.0).

To account for fluctuations in request volume, the system uses the greater of the previous epoch's total requests and the current epoch's total requests to determine the validation frequency.

**Reputation management**

- Inactivity: If an executor does not participate in a Sprint but later returns, their reputation remains unchanged.
- Fraud detection: Reputation is only reduced if the executor is caught engaging in fraudulent activities.
- Long-term inactivity: If an executor remains inactive for over a month, their reputation is deleted to avoid storing unnecessary data. Upon returning, the executor's reputation resets to zero.

**Reward sharing mechanism**

Executors share a portion of their rewards with validators as an incentive for validation efforts. The amount shared depends on the executor's reputation and the corresponding validation frequency.

- Each time an executor is validated, the reward is split equally among the executor and all validating validator(s).
- Executors with low reputations may be validated multiple times (or not at all), but the average validation rate will be once per task.
- If multiple validators check the same task, the rewards are divided among them.
- High-reputation executors, who are validated only 1% of the time, benefit from retaining a larger share of their rewards:
    - In 99% of cases, they keep 100% of the reward.
    - In 1% of cases, they share 50% of the reward.
- If an executor's reputation decreases, they revert to a higher validation frequency, increasing their reward-sharing obligation.

**Handling unfinished inferences**

To ensure users are refunded for unfinished inferences, the system employs a mechanism for tracking and expiring hanging inferences:

- Unfinished inferences are stored in a separate storage area.
- Once an inference is completed, it is moved to the main inference storage.
- Every X blocks (e.g., X = 100), the system scans the unfinished pool for hanging inferences.

Any inference older than Y blocks (e.g., Y = 20) is considered hanging and is expired, triggering a refund to the user.

# Appendix E. Decentralized Synchronization and Validation in Large-Scale Distributed Learning

A scalable framework for decentralized AI training requires efficient synchronization, trustless validation, and support for massive model sizes. Foundational works, such as DiLoCo, have addressed some of these challenges by enabling low-communication distributed training. This appendix defines a new mechanism that makes it possible to train truly decentralized large foundation models collaboratively across untrusted distributed Hosts, making it a critical enabler of the Gonka network's core mission.

### The DiLoCo Mechanism

In classical distributed learning, Hosts typically synchronize after every training step, leading to enormous communication overhead and waiting times as models grow larger. DiLoCo [6] challenges this approach by introducing a different synchronization strategy. Instead of continuous parameter exchange, DiLoCo synchronizes the model's parameters only periodically (e.g., every 1000 training steps).

What enables DiLoCo to synchronize less frequently is its bi-level optimization process, building upon foundational work in federated learning [8][9]. Each Host performs multiple iterations independently using optimizers such as AdamW, while the outer optimization loop, inspired by Federative Averaging, synchronizes local models every N steps. This ensures efficient use of local compute resources. The outer optimization loop, unique to DiLoCo, introduces advanced techniques like Nesterov momentum to aggregate updates across Hosts periodically. By carefully pairing these optimizers, DiLoCo achieves both stability in global updates and flexibility in local computation.

The outcome is a learning environment capable of operating effectively across geo-distributed networks. By drastically lowering communication overhead, DiLoCo enables more efficient large-scale training than traditional distributed approaches which require every step synchronization. DiLoCo's scaling laws [10] show predictable performance improvements as model sizes grow, making it particularly suitable for training foundation models. Practical implementations of DiLoCo can train 30-50 billion parameter models using configurations of 8xH100 GPUs servers.

### Transition from Centralized to On-Chain Management

Originally, DiLoCo implementations have relied on a centralized Host that managed the synchronization schedule, rendezvous, and assigned ranks to the Hosts. This structure introduced a single point of failure and potential trust concerns. To resolve these issues, our approach replaces the centralized Host with a decentralized on-chain management framework.

Critical processes such as rendezvous, rank assignment, and synchronization coordination are now handled through distributed blockchain mechanisms. This ensures that no single entity controls the training process, reinforcing trustlessness and resilience to probable Host failure/disconnect.

By incorporating on-chain management, the training process becomes more democratic and transparent, preventing any single authority from unilaterally affecting outcomes, manipulating global parameters or disrupting the whole training process.

### Proof-of-Learning and Trustless Collaboration

In a decentralized training environment, all Hosts are inherently untrusted. This fundamental assumption drives our approach to validation. We need mechanisms that allow Hosts to verify others are not misrepresenting their work or manipulating the learning process.

Our approach incorporates elements inspired by "Proof of Learning" [7], ensuring that Hosts who falsify training will eventually be detected and penalized. The system employs probabilistic validation whereby Hosts are subject to verification at pseudo-randomly selected intervals. This approach strikes a balance between comprehensive oversight and computational efficiency.

**State Preservation**

At predetermined, pseudo-randomly selected steps during training, each participating Host preserves critical artifacts to verify training steps: previous weights, current weights, and optimizer states (including momentum terms and adaptive learning rates). These preserved states serve as evidence of legitimate training activities and create a verifiable proof of the learning process.

Hosts commit hashes of their preserved states to the blockchain. These hash commitments provide timestamped, tamper-evident records of training progress while requiring minimal on-chain storage. The actual training artifacts remain off-chain but have to be provided on demand during validation procedures.

**Validation Process**

The validation process operates through randomly selected validators drawn from Hosts not currently engaged in the model training process. These validators periodically request the underlying data from training Hosts, verify that the hash matches the on-chain commitment, and confirm that weight changes represent legitimate training iterations based on the provided data. Should discrepancies arise—for instance, if a Host attempts to submit falsified weights—validators flag these issues on-chain, preserving network integrity.

Crucial to our approach is the understanding that validators themselves cannot be inherently trusted. To address this, validation results are permanently recorded on-chain, allowing any Host to independently verify these assessments at any time. We implement a multi-layered verification system where validators are themselves subject to oversight through honeypot traps (intentionally flawed submissions designed to test validator diligence) inspired by [7] and re-validation by other validators.

This mechanism ensures a trustless, verifiable training environment. Malicious actors attempting to fake progress or insert incorrect weights would be promptly caught, upholding data integrity and fostering collaboration.

**Unlocking Decentralized AI Training at Scale**

DiLoCo's reduced synchronization frequency, on-chain management, and proof-of-learning-based validation tackle bandwidth constraints, centralization risks, and trust issues. However, to enable decentralized training of truly large models, we must also solve the scalability problem. As models grow to hundreds of billions of parameters, we cannot reasonably expect all Hosts to train the full model on their servers, this would be too harsh a constraint and would prevent many potential Hosts from joining the training. For now, we still expect Hosts to be able to store the full model on one server for inference purposes to maintain performance,

To address this challenge, various sharding approaches have been developed that divide models into distinct portions distributed across multiple Hosts. This allows each Host to store and train only a subset of the model's parameters. This concept has been explored in several notable works. GShard [11] introduced efficient scaling techniques for trillion-parameter models using conditional computation and model parallelism in distributed Mixture-of-Experts (MoE) training; Decentralized Mixture-of-Experts [12] proposed decentralized training methods for MoE architectures, introducing a framework for training large models using unreliable consumer-grade hardware; and DiPaCo [13] presented techniques for distributed parameter coordination in large-scale training. Paths of shared 150M-parameter modules are trained using DiLoCo's synchronization (outer-gradient aggregation every 100 steps), achieving 45% faster training than dense baselines. The resulting model combines 256 paths that share some of the parameters, and has a total size of 20B parameters, requiring to store only 150M on each Host during training.

Our framework allows combining the DiLoCo with these sharding approaches. Thus the maximum feasible model size grows significantly because each Host handles only a portion of the model. During training, nodes maintain only the parameters for their assigned portions and share updates related to these components when synchronization events occur. With this combined approach, training models at large scales becomes viable, including models comparable to DeepSeek's R1 with its 671B parameters and larger as the computational and memory burden is distributed across the network rather than concentrated on individual nodes.

By integrating efficient synchronization mechanisms, on-chain management, validation, and model sharding techniques, we create a comprehensive framework for decentralized, large-scale AI training. This approach opens possibilities for community-driven development of sophisticated AI systems through collaborative, distributed efforts.

## References


1. Joe-Wong, Carlee and Sen, Soumya. Pricing the Cloud: Resource Allocations, Fairness, and Revenue. https://www.andrew.cmu.edu/user/cjoewong/Cloud_WITS.pdf
2. Jones, Mike. “Microsoft Azure Negotiation Strategy – Large Deals.” https://www.uscloud.com/blog/microsoft-azure-negotiation-strategy-large-deals/
3. Wang, Sarah and Casado, Martin. “The Cost of Cloud, a Trillion Dollar Paradox.” https://a16z.com/the-cost-of-cloud-a-trillion-dollar-paradox/
4. Peters, Keaton. “US Government Launches New Attempt to Gather Data on Electricity Usage of Bitcoin Mining.” https://insideclimatenews.org/news/11072024/us-energy-information-agency-bitcoin-mining-electricity-usage-data/
5. Nakamoto, Satoshi. “A Peer-to-Peer Electronic Cash System.” https://bitcoin.org/bitcoin.pdf.
6. Douillard, Arthur, et al. "Diloco: Distributed low-communication training of language models."
7. Jia, Hengrui, et al. "Proof-of-learning: Definitions and practice."
8. McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data."
9. Reddi, Sashank, et al. "Adaptive federated optimization." International Conference on Learning Representations.
10. Charles, Zachary, et al. "Communication-Efficient Language Model Training Scales Reliably and Robustly: Scaling Laws for DiLoCo."
11. Lepikhin, Dmitry, et al. "Gshard: Scaling giant models with conditional computation and automatic sharding."
12. Ryabinin, Max, and Anton Gusev. "Towards crowdsourced training of large neural networks using decentralized mixture-of-experts."
13. Douillard, Arthur, et al. "Dipaco: Distributed path composition."
14. Bitcoin Block Reward Halving Countdown. https://www.bitcoinblockhalf.com/.
15. “How The Merge impacted ETH supply”. https://ethereum.org/en/roadmap/merge/issuance/