Allora: a Self-Improving, Decentralized Machine Intelligence Network

J. M. Diederik Kruijssen,[1] Nicholas Emmons,[1] Kenneth Peluso,[1] Faisal Ghaffar,[1] Alexander Huang[1] & Tyler Kell[1]
[1]*Allora Foundation*

Litepaper v2024.3.20

Abstract

Recent advances in data access and computing power have enabled the first forms of machine intelligence capable of offering meaningful insights. However, the tremendous resources required have caused these solutions to be closed and siloed by industry monoliths. To achieve the full potential of machine intelligence, the data, algorithms, and participants must be maximally connected. Network solutions are needed, and the decentralized nature of blockchain technology is ideal for solving this problem. We introduce Allora, a self-improving, decentralized machine intelligence network that surpasses the capabilities of its individual participants by design. Allora achieves this through two main innovations. First, it lets network participants forecast each other's performance under the current conditions, thereby creating a form of context-awareness that enables the network to achieve the best inferences under any circumstances. Second, it introduces a differentiated incentive structure that rewards network participants for their unique contribution to the network goal, tailored to their specific task and purpose, avoiding any distracting incentives. We show that these innovations make Allora's inferences considerably more accurate than before. With its versatility and accessibility, Allora paves the way for machine intelligence to become fully commoditized and integrated with the economy, technology, and society.

1 The Problem: Siloed Machine Intelligence

Machine intelligence represents the ability of a machine to learn, improve, and work proactively through artificial intelligence (AI) and machine learning (ML). It is built by conditioning advanced algorithms on large volumes of informative data using state-of-the-art computational infrastructure. We live in the information age, where major advances in data generation, processing, and availability are combined with revolutionary computational capacity. These developments have unlocked major advances in machine intelligence, capable of providing insights beyond the reach of human inference, across a wide variety of use cases (e.g. Jacobs et al., 1991; Shazeer et al., 2017; Lightman et al., 2023; OpenAI, 2023). As such, machine-generated intelligence is becoming a valuable resource and a popular commodity.

The coordination of resources needed to build machine intelligence, coupling data, algorithms, and computational power, has naturally favored industry monoliths. These now hold the keys to the best performing forms of automated inference ever created. This not only monopolizes the ability to control, direct, and use this revolutionary technology, but also creates a lack of transparency and a major barrier to entry for developers and users. Fundamentally, this siloed approach also violates a key prerequisite for building the best possible form of machine intelligence: to maximize the number of connections across a network in which diverse data sets and algorithms can be freely coupled, so that the most relevant insights can be obtained (e.g. Vaswani et al., 2017; Bzdok et al., 2019).

The problem at hand naturally requires network solutions that create a form of swarm intelligence, connecting a large number of data sets and inference algorithms (e.g. McMahan et al., 2017). The decentralized nature of blockchain technology lends itself ideally to solving this problem, allowing bespoke incentive structures that align the interests of network participants with those of the network, and facilitating the exchange of value needed to directly support an intelligence economy (e.g. Nakamoto, 2008; Buterin, 2014).

Existing blockchain initiatives that attempt to solve the machine intelligence problem are sub-optimal. Some well-known solutions adopt such a strictly decentralized philosophy that they are unable to support different incentive structures for different actors within the network (e.g. Craib et al., 2017; Rao et al., 2021; Steeves et al., 2022). Additionally, these solutions often reward network participants using traditional blockchain objectives such as their integrated historical reputation, making them struggle to achieve context-aware intelligence that provides the best solution in the specific context where an inference is needed.

Allora is a self-improving, decentralized machine intelligence network that overcomes these fundamental challenges. Allora is built on the desire to create a world where machine intelligence supports and improves humanity by offering unique and actionable insights that outperform all other forms of inference. In this world, machine intelligence is openly accessible and transparent, inviting contributions from anyone with data or algorithms that improve the network.

Allora acknowledges that different roles within the network require different incentive structures, and that selecting the best inference across a network of participants often depends on contextual details that themselves may require machine intelligence to be identified. As was recognized by Buterin (2024), "*there is a need for a higher-level game which adjudicates how well the different AIs are doing, where AIs can participate as players in the game.*" By recognizing these fundamental aspects of machine intelligence and adopting innovative solutions to address them, the Allora network

returns inferences that outperform the strongest network participant by definition, yet rewards each of them fairly for their contribution towards achieving this goal. This solves a fundamental challenge in decentralized learning and machine intelligence.

While the involvement of industry monoliths in the quest for machine intelligence has made it seem like a winner-takes-all race, the arrival of Allora now introduces a fully decentralized solution that outperforms any individual contributor. In this way, the end user is the winner, and machine intelligence belongs to everyone.

2 Allora: Self-Improving, Decentralized Machine Intelligence

The Allora network is a state-of-the-art protocol that uses decentralized AI and ML to build, extract, and deploy AI predictions or inferences among its participants. It offers a formalized way to obtain the output of ML models in blockchain networks of virtual machines (VMs) and to reward the operators of AI nodes who create these inferences. In this way, Allora bridges the information gap between data owners, data processors, AI models, and the end users or consumers who have the means to execute on these insights.

The AI agents within the Allora network use data and algorithms to generate inferences, which they then broadcast across a peer-to-peer network. A second set of agents evaluates the quality of these inferences using a network consensus mechanism. The network then uses these assessments to generate a single collective inference. Over time, this network inference outperforms any individual AI agent by construction, thanks to the unique innovations of the Allora design. The network distributes rewards to the agents according to their individual contributions to the network inference. This carefully designed incentive mechanism enables Allora to continually learn and improve, adapting to each inference problem as it evolves.

Allora introduces a variety of innovations that represent important steps towards our goal of achieving self-improving, decentralized machine intelligence. The network has been designed following a modular philosophy that recognizes the need for bespoke solutions that best satisfy local boundary conditions. The main defining characteristics are Allora's **context awareness** and its **differentiated incentive structure**. After first providing a brief overview of the network structure, we expand on these key concepts further below.

The Allora ecosystem is built on a **hub chain** that coordinates the macroeconomics of the network, including the tokenomics of the network's native ALLO token and the emission of subsidies and rewards, as well as other coordination tasks. To help organize the problems that the Allora network can solve, we introduce the concept of **topics**. These are sub-networks within which network participants collaborate to generate inferences and earn rewards. Each topic contains a short rule set that governs the interaction between the topic participants, including the **target variable** and the **loss function** that needs to be optimized by the topic network. The discussion of this paper focuses mainly on the topic infrastructure, which is illustrated schematically in Figure 1.

After any of the Allora network participants create a topic, the participants can perform a variety of different roles.

1. **Workers** provide AI-powered inferences to the network. There exist two kinds of inference that workers produce within a topic. The first refers to the target variable that the network topic is generating (**inference** in Figure 1). The second refers to the **forecasted losses** of the inferences produced by other workers (**forecasting** in Figure 1). These forecasted losses represent an expectation of performance rather than a reported performance, and represent the fundamental ingredient that makes the network context-aware, as they provide insight into the accuracy of a worker under the current conditions. For each worker, the network uses these worker-forecasted losses to generate a forecast-implied inference that combines the original inferences of all workers. A worker can choose to provide either or both types of inference, and receives rewards proportional to its unique contribution to the network accuracy, both in terms of its own inference and its forecast-implied inference.

2. **Reputers** evaluate the quality of the inferences and forecast-implied inferences provided by the workers. This is done by comparing the inferences with the ground truth when it becomes available. Reputers are the source of economic security in the network, as a reputer receives rewards proportional both to its stake and the consensus between its evaluations and those of other reputers.

3. **Consumers** request inferences from the network. A consumer uses tokens to pay for these inferences.

The interactions between these participants are coordinated by the Allora topic rule set (referred to as **topic coordinator** in Figure 1). Together, they represent the ingredients needed to achieve a self-improving, decentralized form of machine intelligence.

3 Allora's Context-Aware and Self-Improving Intelligence Mechanism

The first of two critical hurdles to achieving decentralized machine intelligence is to optimally combine the inferences produced by network participants. This means that the network must recognize both the historical and context-dependent accuracy of these inferences. Especially the latter of these requirements has posed a challenge: most initiatives attempting

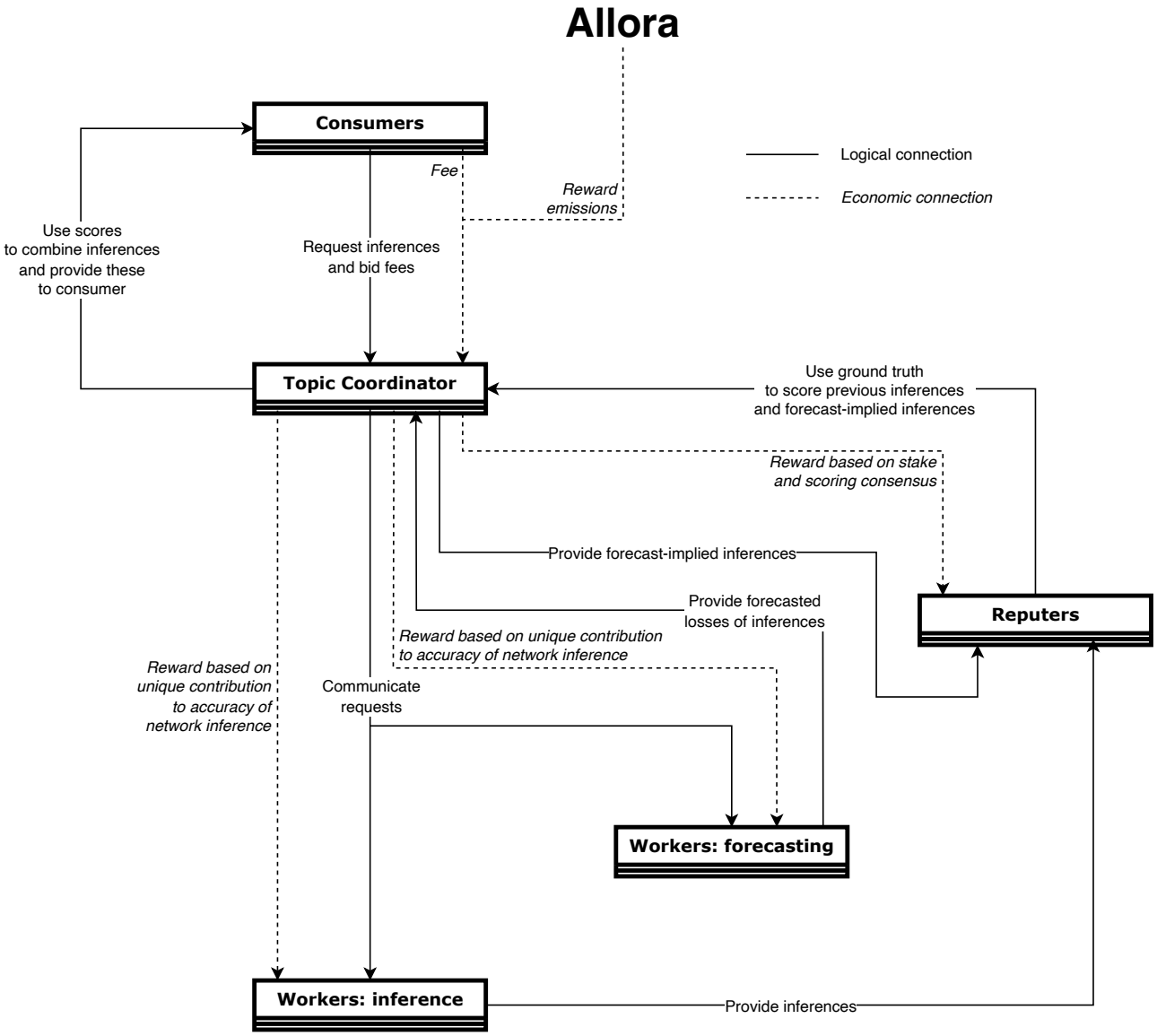

Figure 1: *Schematic representation of an Allora 'topic', which is a sub-network within the Allora ecosystem characterized by a specific AI target and loss function. Topics help organize the problems that Allora is solving, and they are used to coordinate the collaboration between network participants. This schematic illustrates the logical and economic interactions between workers, reputers, and consumers. The 'topic coordinator' represents the rule set that Allora uses to coordinate these interactions.*

to build machine intelligence exclusively rely on cumulative historical reputation to combine inferences while ignoring deterministic variations in their accuracy, which prohibits the network to be context-aware. Allora overcomes this challenge through a process called **Inference Synthesis**, which we describe in this section.

In describing the functionality of Allora's intelligence mechanism, we assume that there exists an online data stream that is used to obtain periodic inferences for a total number of time steps or epochs N_{e}, where $i \in \{1, \ldots, N_{\mathrm{e}}\}$ indicates the epoch. Furthermore, we consider a system consisting of N_{w} workers that provide N_{i} inferences during the inference task and N_{f} inferences during the forecasting task. For simplicity, we assume full participation ($N_{\mathrm{w}} = N_{\mathrm{i}} = N_{\mathrm{f}}$), but this need not be the case. During the inference task, each worker $j \in \{1, \ldots, N_{\mathrm{i}}\}$ produces an inference I_{ij} for the target variable of the topic, using its own data set D_{ij} and model M_{ij}:

$$I_{ij} = M_{ij}(D_{ij}). \tag{1}$$

During the forecasting task, each worker $k \in \{1, \ldots, N_{\mathrm{f}}\}$ produces an inference for the forecasted loss L_{ijk} of the inference I_{ij} produced by worker j, using its own (potentially augmented) data set D_{ijk} and model M_{ijk}:

$$L_{ijk} = M_{ijk}(D_{ijk}), \tag{2}$$

where M_{ijk} should be defined such that $L_{ijk} > 0$ (e.g. by using powers of 10 if needed).

Notation. We employ a subscript notation to indicate the association of variables with different components of the network. The notation is based on a combination of indices (i, j, k, l, m), each denoting a specific element or task within the network. The last index in the sequence denotes the network element or task with which the variable is associated. Specifically, when the last index is:

- i: The variable is associated with the network itself ('topic coordinator' in Figure 1).
- j: The variable is associated with a worker carrying out an inference task, in which it infers the topic's target variable ('worker: inference' in Figure 1).
- k: The variable is associated with a worker carrying out a forecasting task, in which it forecasts the loss of another worker's inference ('worker: forecasting' in Figure 1).
- l: The variable is associated with a worker carrying out either the inference or the forecasting task, and has been obtained by appending the arrays associated with each of these individual tasks.
- m: The variable is associated with a reputer, which calculates and reports the loss of an inference of the topic's target variable ('reputer' in Figure 1).

Note that any of these indices can be operated on (e.g. subtraction to refer to earlier time steps), and thus we equate e.g. $X_{i,j} \equiv X_{ij}$ occasionally to improve legibility. Our notation formalism also implies that e.g. X_i, X_{ij}, X_{ik}, and X_{ijk} are all different variables. Finally, we use caligraphic script to refer to variables that represent a ground truth, such as a loss or a regret.

The losses forecasted by workers during the forecasting tasks reflect how accurate worker k expects the inference I_{ij} to be, given the contextual information D_{ijk}. This correlation between performance and context is the critical ingredient that makes Allora context-aware. The forecasted losses are used to obtain the forecast-implied inference of the topic's target variable through a weighted average:

$$I_{ik} = \frac{\sum_j w_{ijk} I_{ij}}{\sum_j w_{ijk}}. \tag{3}$$

To calculate the weights w_{ijk}, we first approximate the forecasted regret R_{ijk} of the network-wide inference I_i to be constructed at the current epoch by subtracting the logarithms of the forecasted losses L_{ijk} from the network loss $\mathcal{L}_{i-1}$ that was reported at the previous time step, which results in:

$$R_{ijk} = \log\left(\frac{\mathcal{L}_{i-1}}{L_{ijk}}\right). \tag{4}$$

A positive regret implies that the inference of worker j is expected by worker k to outperform the network's previously reported accuracy, whereas a negative regret indicates that the network is expected to be more accurate. The regrets are converted to weights following a gradient descent approach, where the weights are set by the gradient of a potential function $\phi_p(x)$:

$$w_{ijk} = \phi_p'(\hat{R}_{ijk}), \tag{5}$$

where we define a simple potential function

$$\phi_p(x) = \ln\left(1 + \mathrm{e}^x\right)^p, \tag{6}$$

where we adopt $p = 2$ as a fiducial value. This potential function is a smoothly differentiable approximation of $\max(0, x)^p$, reflecting the intention that the workers providing inferences that are expected to have negative regrets should be assigned negligible weights. The resulting gradient is

$$\phi_p'(x) = \frac{p \ln\left(1 + \mathrm{e}^x\right)^{p-1} \mathrm{e}^x}{1 + \mathrm{e}^x}. \tag{7}$$

Before the forecasted regret is passed to the potential function, it is normalized as

$$\hat{R}_{ijk} = \frac{R_{ijk}}{|\max_j(R_{ijk})|}, \tag{8}$$

where $\max_j$ indicates taking the maximum over all $j \in \{1, \ldots, N_\mathrm{i}\}$. Normalization by the absolute value of the maximum forecasted regret restricts $\hat{R}_{ijk}$ to the range $(-\infty, \pm 1]$, with the sign of the upper bound depending on whether the maximum regret is positive or negative. This normalization ensures that the network always obtains a non-zero weight for at least one inference, which when all forecasted regrets are negative may not happen otherwise, while maintaining the intended distribution of similarly-low weights for all-negative forecasted regrets and steeply increasing weights for positive regrets. Using these definitions, the network has access to a total of $N_\mathrm{i} + N_\mathrm{f} \leq 2N_\mathrm{w}$ inferences, with I_{ij} being the original set of inferences from the inference task, and I_{ik} being the set of context-aware forecast-implied inferences from the forecasting task.[1]

[1] Recall that a worker may choose to engage (or not) in any of the tasks, so $2N_\mathrm{w}$ inferences represents an upper limit to the total number. For simplicity, we assume here that all workers perform all tasks, but the design outlined in this paper does not require this assumption to be satisfied.

The network concludes the Inference Synthesis and obtains a network-wide inference by combining the inferences I_{ij} and the forecast-implied inferences I_{ik} through a procedure similar to the one described above. We first define a new variable I_{il} that appends both sets of inferences in a new array with $l \in \{1, \ldots, N_\text{i} + N_\text{f}\}$. The network inference is then defined as

$$I_i = \frac{\sum_l w_{il} I_{il}}{\sum_l w_{il}}. \tag{9}$$

This time, the weights are not set by a forecasted regret, but by the actual regret for each worker task obtained during the previous time step $\mathcal{R}_{i-1,l}$ (which we will define in Equation 15), as

$$w_{il} = \phi'_p(\hat{\mathcal{R}}_{i-1,l}), \tag{10}$$

with

$$\hat{\mathcal{R}}_{i-1,l} = \frac{\mathcal{R}_{i-1,l}}{|\max_l(\mathcal{R}_{i-1,l})|}, \tag{11}$$

analogously to Equation 8.

In addition to the network inference I_i of Equation 9, the network generates a wide variety of secondary inferences that are based on various subsets of the inferences obtained during the inference and forecasting tasks. These secondary inferences are used to derive the confidence intervals of the network inference, and to quantify the unique contribution of workers through their inference and forecasting tasks to improving the accuracy of the network inference, which is used to determine their reward allocations (see §4). These secondary inferences include:

1. A 'naive' network inference I_i^-, which omits all forecast-implied inferences from the weighted average in Equation 9 by setting their weights to zero. The naive network inference is used to quantify the contribution of the forecasting task to the network accuracy, which in turn sets the reward distribution between the inference and forecasting tasks.

2. A 'one-out' network inference I_{li}^-, which omits a single inference or forecast-implied inference l from the weighted average in Equation 9. If an inference from the inference task is omitted (I_{ij}), the forecast-implied inferences (I_{ik}) are updated accordingly before calculating I_{li}^-. The one-out network inferences represent an approximation of Shapley (1953) values and are used to quantify the individual contributions of workers to the network accuracy, which in turn sets the reward distribution between workers. The one-out network inferences are also used to calculate confidence intervals on the network inference I_i.

3. A 'one-in' naive network inference I_{ki}^+, which adds only a single forecast-implied inference I_{ik} to the inferences from the inference task I_{ij}. As such, it is used to quantify how the naive network inference I_i^- changes with the addition of a single forecast-implied inference, which in turn is used for setting the reward distribution between workers for their forecasting tasks. The one-in naive network inference better differentiates between forecast-implied inferences than their associated one-out inferences, because there exists some redundancy between multiple forecasting tasks and omitting a single one need not negatively impact the network inference. After all, the forecasting tasks can only draw from a finite number of original inferences and forecast-implied inferences may sometimes be mutually exchangable. This redundancy is desirable from a decentralization perspective and should not be disincentivized by exclusively using the one-out network inference to reward workers for the forecasting task. Therefore, we additionally use the one-in naive network inference to quantify each worker's individual contribution.

All inferences generated by the network are collected as

$$\mathbf{I}_i = \{I_i, I_{ij}, I_{ik}, I_i^-, I_{li}^-, I_{ki}^+\}, \tag{12}$$

and are evaluated by the reputers. The reputers obtain a ground truth of the target variable $\mathcal{Y}$ when it becomes available (for simplicity, we assume this is delayed by one time step) and compare it with each of the inferences by calculating the topic's loss function Q.

$$\begin{aligned}
\mathcal{L}_{im} &= Q(I_{i-1}, \mathcal{Y}_{i-1}), \\
\mathcal{L}_{ijm} &= Q(I_{i-1,j}, \mathcal{Y}_{i-1}), \\
\mathcal{L}_{ikm} &= Q(I_{i-1,k}, \mathcal{Y}_{i-1}), \\
\mathcal{L}_{im}^- &= Q(I_{i-1}^-, \mathcal{Y}_{i-1}), \\
\mathcal{L}_{lim}^- &= Q(I_{l,i-1}^-, \mathcal{Y}_{i-1}), \\
\mathcal{L}_{kim}^+ &= Q(I_{k,i-1}^+, \mathcal{Y}_{i-1}).
\end{aligned} \tag{13}$$

It is left to the individual reputer to decide whether the loss reported at time step i also includes some record of the historical loss, e.g. by using an exponential moving average. Doing so introduces a free parameter α_m that controls the relative weights of the historical and current losses. Because reputers are rewarded based on their consensus (see §4), this

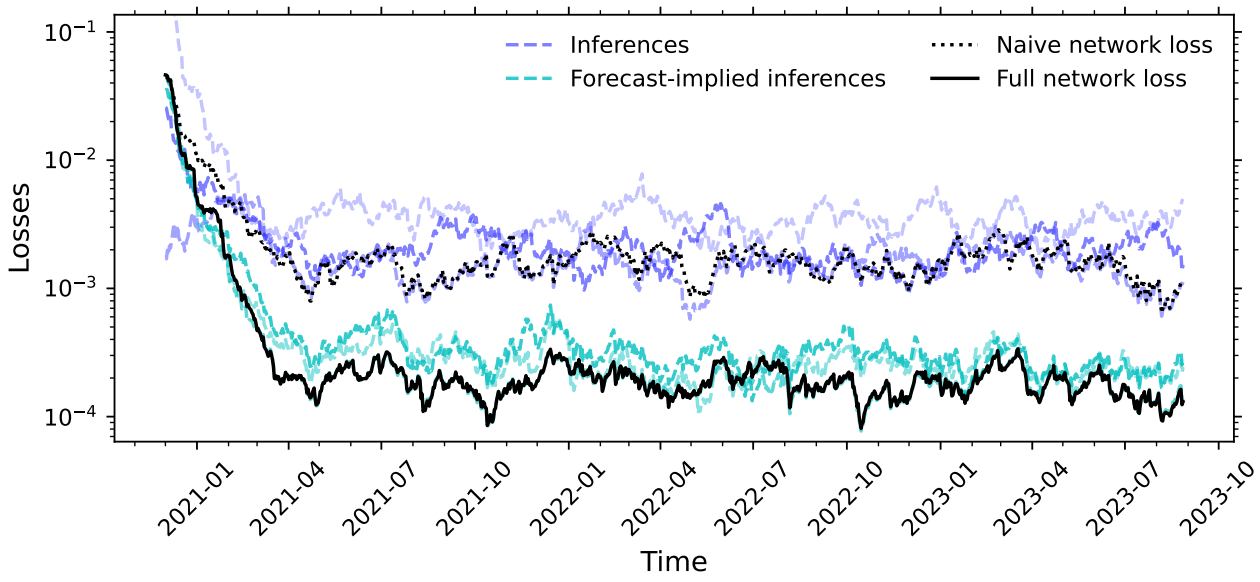

Figure 2: *Demonstration of Allora's self-improving intelligence and the accuracy improvement due to its context-aware Inference Synthesis mechanism. The dotted black line shows the naive network loss as a function of time, which is obtained by combining individual inferences (blue) without context awareness. The solid black line shows an order of magnitude improvement in loss thanks to the introduction of the forecasting task (cyan), which correlates performance and context by letting workers forecast each other's losses under the current conditions.*

introduces a game-theoretical aspect to the reputer task. It is expected that reputers achieve consensus on the choice of α_m that optimizes the accuracy of the network inference.

Each reputer has a stake S_{im}, and the losses reported by the reputers are combined through a stake-weighted average of the log-loss:

$$\log \mathcal{L}_i = \frac{\sum_m S_{im} \log \mathcal{L}_{im}}{\sum_m S_{im}}, \tag{14}$$

where we have avoided specifying all six variations listed in Equation 13 for brevity. The resulting losses are used by the network to calculate the corresponding regrets, which set the weights of the network inference as specified in Equation 9–Equation 11. The regrets are calculated according to an exponential moving average with a fiducial parameter $\alpha \in (0, 1]$:

$$\mathcal{R}_{il} = \alpha \log \left(\frac{\mathcal{L}_i}{\mathcal{L}_{il}} \right) + (1 - \alpha) \mathcal{R}_{i-1,l}. \tag{15}$$

We adopt $\alpha = 0.1$ as the fiducial value, which provides a reasonable balance between historical performance and recency. Corresponding regrets are also calculated for the losses of the secondary inferences listed in Equation 13, to enable evaluating Equation 10 for each of these.

Figure 2 illustrates the performance of the Allora architecture. We consider a simple numerical experiment using mock data, where three workers generate inferences that predict the ground truth with some specified accuracy. At each time step, the error of a different worker is temporarily decreased by a factor 0.3 to mimic context-dependent outperformance. The workers also forecast each other's losses with different degrees of context-sensitivity, i.e. their ability to anticipate this temporary outperformance. Figure 2 shows the losses of the resulting forecast-implied inferences, together with the losses of the original inferences, the loss of the naive network inference, and the loss of the complete network inference. On average, the naive network is as good as or better than the best inference. However, the forecast-implied inferences are even more accurate, and the network-wide inference outperforms all other inferences on average. We find that the addition of the forecasting task greatly improves the network accuracy, even in cases where the accuracy of the workers performing the forecasting task is lower than during the inference task. Even a moderate contextual awareness of when inferences are typically more accurate seems to be sufficient to yield a net improvement of the network accuracy.

With the presented design, Allora optimally combines the inferences produced by the network participants through its Inference Synthesis mechanism. This is achieved by recognizing both the historical and context-dependent accuracy of the inferences. The key ingredient is the introduction of the forecasting task, which correlates performance and context by letting workers forecast each other's losses under current conditions.

4 Allora's Differentiated Incentive Structure

4.1 Reward distribution among individual network participants

The second critical hurdle to achieving decentralized machine intelligence is creating custom incentive structures that appropriately reward different actions within the network. Workers should be rewarded for their inference and forecasting tasks according to their unique contribution to the network. Fundamentally, there is no reason why this reward should depend on a monetary commitment such as a stake; in fact, stake-dependent rewards distract from their single objective of maximizing the network accuracy. By contrast, reputers must reach a form of consensus on their reported losses and should be rewarded for their proximity to that consensus. Because the network should reward consensus among reputers, their rewards can follow the common practice in decentralized systems of depending on the stake. By reporting on the performance of workers and thereby influencing their reward allocation too, reputers' stakes provides the economic security for the entire topic.

Common ways to quantify the unique contribution of participants to an end result include the Shapley (1953) value, Fisher (1922) information score, Banzhaf (1965) Power Index, and many others. The computational cost of these metrics is often high due to their reliance on large permutation sets, which can be prohibitive in a decentralized network setting. Therefore, Allora adopts a simple approximation of the Shapley values to score worker performance. For the inference task, we define the performance score as the log-loss difference between the one-out inference and the network inference, where the inference provided by a worker during the inference task is omitted from the network inference:

$$T_{ij} = \log \mathcal{L}_{ji}^{-} - \log \mathcal{L}_i, \tag{16}$$

where j in the subscript indicates that we only consider one-out losses of inferences from the inference task. Recall that these one-out losses include the secondary impact of omitting an inference on the forecast-implied inferences by recalculating these (see §3). The performance score T_{ij} is positive if the removal of an inference would increase the network loss, and is negative if its removal would decrease the network loss.

The worker performance during the forecasting task can be scored similarly, but requires additional information from the one-in inferences introduced above. The forecasting task is comparatively redundant, i.e. in order to function well, a topic requires only one worker with reasonable context awareness to provide forecasted losses. As a result, removing any individual worker from the forecasting task may not noticeably impact the network inference loss, but the redundancy between workers is desirable (and should be rewarded) from a network perspective. A complete Shapley value calculation would remedy this problem by considering all possible permutations of workers, but at a prohibitive computational cost. Allora sidesteps this issue by adding only a single, information-rich permutation per worker, where the forecast-implied inference of that worker is added to the naive network inference to quantify its individual impact. The worker performance score then becomes a combination of the one-out score and the one-in score:

$$T_{ik} = (1 - f^{+})T_{ik}^{-} + f^{+}T_{ik}^{+}, \tag{17}$$

where we define the one-out score analogously to Equation 16:

$$T_{ik}^{-} = \log \mathcal{L}_{ki}^{-} - \log \mathcal{L}_i, \tag{18}$$

and the one-in score as:

$$T_{ik}^{+} = \log \mathcal{L}_{i}^{-} - \log \mathcal{L}_{ki}^{+}. \tag{19}$$

It is easy to verify that these definitions satisfy the required directionality of the scores, i.e. the one-out score increases if the removal of a forecast-implied inference would increase the network inference loss $\mathcal{L}_i$, and the one-in score increases if the addition of a forecast-implied inference would decrease the naive network inference loss $\mathcal{L}_i^{-}$. The weight of both terms in Equation 17 is parameterized using f^{+}, which represents the fraction of permutations in a binomial experiment in which a worker appears solo, i.e.

$$f^{+} = \frac{1}{2^{N_{\mathrm{f}}}}, \tag{20}$$

where N_{f} is the number of workers providing forecasted losses during the forecasting task.

The scores obtained in Equation 16 and Equation 17 facilitate the distribution of rewards to workers for inference and forecasting tasks. Given some total reward allocated to each of these tasks per time step (U_i and V_i, both specified in detail below), we use the scores to calculate the rewards received by each individual worker. For the inference task, these are defined as

$$U_{ij} = U_i \frac{M(T_{ij})}{\sum_j M(T_{ij})}, \tag{21}$$

and for the forecasting task, these are

$$V_{ik} = V_i \frac{M(T_{ik})}{\sum_k M(T_{ik})}. \tag{22}$$

Here, M is a mapping function that maps scores to reward fractions and the division by the sum ensures normalization of the reward fractions to unity. The mapping function must satisfy a number of simple requirements. It must reward positive scores and attribute negligible reward to negative scores. It must also be agnostic to the absolute scale of the scores, so that the inference and forecasting tasks are compensated according to a similar differentiation between contributions. Finally, it must accept a free parameter that can be used to control the spread in reward fractions, because this allows the network to influence the (de)centralization of the rewards if needed. The simplest functional form that satisfies these requirements is

$$M(T) = \phi_p \left[\frac{T}{\sigma(T)} \right], \tag{23}$$

where $\sigma(T)$ represents the standard deviation of all scores over the ΔN most recent time steps. The use of the potential function ϕ_p (see Equation 6) ensures that only positive scores receive significant rewards, while negative scores receive a small reward to acknowledge the contribution to decentralization. Dividing by the standard deviation of the scores ensures a similar differentiation between contributions, irrespective of the absolute scale of the score. The parameter p associated with the potential function controls the spread of the rewards. As a fiducial value we adopt $p = 1.5$ for the inference and forecasting tasks, and $\Delta N = 10$ for the time window over which the standard deviation $\sigma(T)$ is evaluated.

As discussed above, reputers require scoring according to their consensus in reporting the ground truth. The naive way of doing this is to calculate the stake-weighted average of all losses reported by a reputer, and to add the rewards to their stakes. However, this creates a runaway effect towards increased centralization, where the reputer with the highest stake has the largest weight in setting the consensus, thereby receiving the highest rewards and further increasing their stake advantage. This can be remedied by using an adjusted stake to set the weight of each reputer when calculating the consensus, where the weight saturates above a certain fraction of the stake. Specifically, Allora assigns an adjusted stake for calculating the consensus as

$$\hat{S}_{im} = 1 - \phi_1^{-1}(\eta)\phi_1 \left[-\eta \left(\frac{N_\mathrm{r} a_{im} S_{im}}{\sum_m a_{im} S_{im}} - 1 \right) \right], \tag{24}$$

where N_r is the number of reputers, ϕ_1 refers to the potential function from Equation 6 with $p = 1$, and a_{im} is a listening coefficient defined below, which falls in the range $[0, 1]$ and represents a weight that the network associates with each individual reputer depending on its historical performance. This function approximates unity for above-average stake fractions ($a_{im} S_{im} / \sum_m a_{im} S_{im} > 1/N_\mathrm{r}$), because the negative argument of the second ϕ_1 term drives it to zero, whereas for below-average stake fractions ($S_{im} / \sum_m S_{im} < 1/N_\mathrm{r}$) it increases linearly from zero at $S_\mathrm{im} = 0$ to approximately unity at $a_{im} S_{im} / \sum_m a_{im} S_{im} = 1/N_\mathrm{r}$. The steepness of the transition is controlled by the parameter η, for which we adopt a fiducial value of $\eta = 20$. This formulation ensures that the consensus calculation is not susceptible to a majority attack, as reputers with above-average stake have equal weight in setting the consensus. The magnitude of their stake influences the rewards received, but cannot be used to further increase their influence in defining the reference point for the reward distribution. This avoids the runaway effect leading to ever-increasing centralization.

With this adjusted definition, we collect all losses reported by a reputer as

$$\mathbf{L}_{im} = \{\mathcal{L}_{im}, \mathcal{L}_{ijm}, \mathcal{L}_{ikm}, \mathcal{L}^-_{im}, \mathcal{L}^-_{lim}, \mathcal{L}^+_{kim}\}, \tag{25}$$

and define the consensus loss vector as

$$\log \mathbf{L}_i = \frac{\sum_m \hat{S}_{im} \log \mathbf{L}_{im}}{\sum_m \hat{S}_{im}}. \tag{26}$$

The reward received by each reputer is set by a combination of its stake and the Euclidean proximity of its reported losses to the consensus loss vector. We score the proximity to consensus as

$$T_{im} = \left[\frac{\|\log(\mathbf{L}_{im}/\mathbf{L}_i)\|}{\|\log \mathbf{L}_i\|} + \epsilon \right]^{-1}, \tag{27}$$

where the first term expresses the relative proximity and $\epsilon = 0.01$ is a small tolerance quantity used to cap reputer scores at infinitesimally close proximities. With these definitions, we can now calculate the listening parameters, which we obtain by gradient descent. As the objective function, we use the stake-weighted total consensus score:

$$T_i = \frac{\sum_m S_{im} T_{im}}{\sum_m S_{im}}. \tag{28}$$

The listening coefficients are initialized at unity and are updated following an iterative process:

$$a_{im} = a_{im} + \lambda \frac{\mathrm{d} \ln T_i}{\mathrm{d} a_{im}}, \tag{29}$$

where λ is the learning rate and $\mathrm{d} \ln T_i / \mathrm{d} a_{im}$ is the relative gradient. The iterative update of Equation 29 is carried out each epoch until the relative gradient reaches $\mathrm{d} \ln T_i / \mathrm{d} a_{im} < 0.001$, or until the maximum of $1/\lambda$ iterations is

reached. Whenever the update of the listening coefficients of Equation 29 decreases the fraction of stake that is listened to below $\sum_m a_{im} S_{im} / \sum_m S_{im} < 0.5$, the differential is instead interpolated to ensure $\sum_m a_{im} S_{im} / \sum_m S_{im} = 0.5$. The resulting listening coefficients carry over into the next epoch, where they are updated using the same process. This gradient descent mechanism enforces consensus and ensures that the network is robust against minority attacks, where reputers incorrectly report on the losses of favored workers. The network learns the listening coefficients a_{im} to ensure that such dishonest reputers are quickly silenced.

Given a total reward allocated to reputers per time step (W_i, specified in detail below), we now calculate the reward per reputer as

$$W_{im} = W_i \frac{(S_{im} T_{im})^p}{\sum_m (S_{im} T_{im})^p}, \tag{30}$$

where the multiplication by stake and consensus score ensures the appropriate dependence of the reward on both quantities, and the parameter p grants the ability to modify the reward spread (we adopt a fiducial value of $p = 1$). Each reputer's reward is added to their stake, so that the stake is constituted by a combination of monetary commitment and historical performance:

$$S_{i+1,m} = S_{im} + W_{im}. \tag{31}$$

This way, poorly performing reputers experience dilution of their stake and weight, whereas accurate reputers can grow their influence. Another major advantage of this form of reward payments to reputers is that the topic is secured by capital in rough proportion to its value.

4.2 Reward division between network tasks

The system described above governs the distribution of rewards among individual participants performing the same task. Next, we define how the total rewards emitted in a given time step E_i are divided among the three classes of tasks within the topic. The differentiation in incentive structures between the classes requires the definition of a new objective function that is relevant in each of the three cases. The common objective of the network is to incentivize decentralization, and this can be quantified for each class of tasks by considering the entropy of the reward distribution among participants performing that task. The entropy increases for larger numbers of participants and for more equal reward distributions.

We define a modified entropy for each class ($\{F_i, G_i, H_i\}$ for the inference, forecasting, and reputer tasks, respectively) to quantify its degree of decentralization:

$$\begin{aligned} F_i &= -\sum_j f_{ij} \ln(f_{ij}) \left(\frac{N_{\text{i,eff}}}{N_{\text{i}}} \right)^\beta, \\ G_i &= -\sum_k f_{ik} \ln(f_{ik}) \left(\frac{N_{\text{f,eff}}}{N_{\text{f}}} \right)^\beta, \\ H_i &= -\sum_m f_{im} \ln(f_{im}) \left(\frac{N_{\text{r,eff}}}{N_{\text{r}}} \right)^\beta, \end{aligned} \tag{32}$$

where we have defined modified reward fractions per class as

$$f_{ij} = \frac{\tilde{U}_{ij}}{\sum_j \tilde{U}_{ij}}, \; f_{ik} = \frac{\tilde{V}_{ik}}{\sum_k \tilde{V}_{ik}}, \; f_{im} = \frac{\tilde{W}_{im}}{\sum_m \tilde{W}_{im}}. \tag{33}$$

Here, the tilde over the rewards indicates that we have applied an exponential moving average

$$\tilde{U}_{ij} = \alpha U_{ij} + (1 - \alpha) U_{i-1,j}, \tag{34}$$

$$\tilde{V}_{ik} = \alpha V_{ik} + (1 - \alpha) V_{i-1,k}, \tag{35}$$

$$\tilde{W}_{im} = \alpha W_{im} + (1 - \alpha) W_{i-1,m}, \tag{36}$$

to enable the entropy to include some record of the recent historical degree of decentralization rather than just the last time step. As before, we adopt a fiducial value of $\alpha = 0.1$.

Finally, the entropy requires modification by the number ratio term in Equation 32, because the increase of the actual entropy with $\ln N$ might otherwise incentivize a sybil attack, in which any class of topic participants to add many copies of themselves to the network in an attempt to boost their entropy and class reward allocation. The number ratio term in Equation 32 has been added to prevent this. It contains an effective number of participants, which is defined as

$$N_{\text{i,eff}} = \frac{1}{\sum_j f_{ij}^2}, \; N_{\text{f,eff}} = \frac{1}{\sum_k f_{ik}^2}, \; N_{\text{r,eff}} = \frac{1}{\sum_m f_{im}^2}. \tag{37}$$

For an equal reward distribution, $N_{\text{eff}} = N$ by definition. For strongly unequal reward distributions, $N_{\text{eff}} \ll N$. With the addition of the $(N_{\text{eff}}/N)^\beta$ term in Equation 32, any sybil attack would necessarily dilute the individual rewards of the

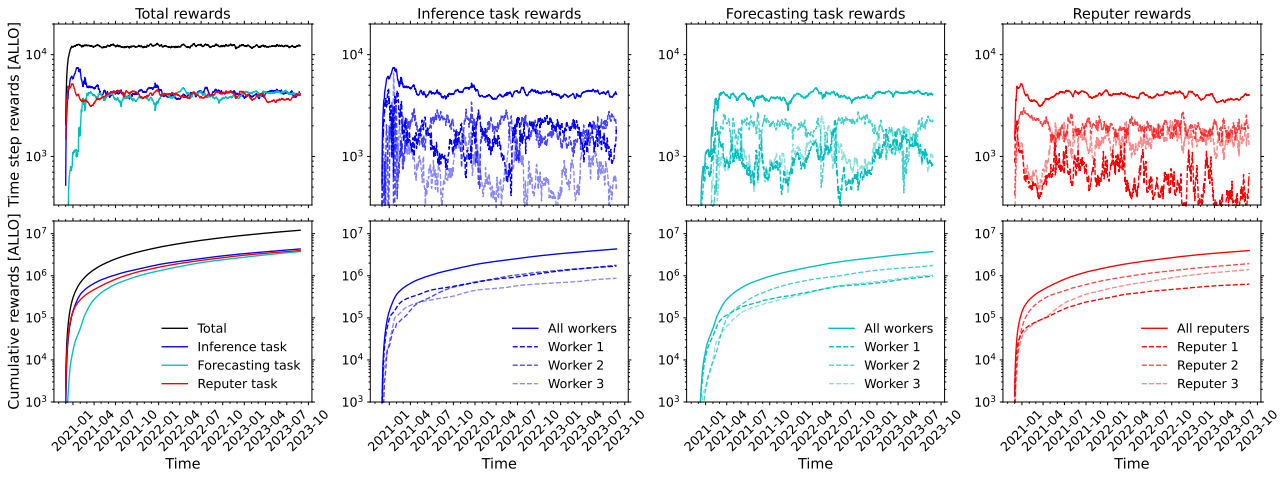

Figure 3: *Demonstration of the incentive structure of the Allora network, showing how rewards are distributed among different network participants over time. The left-hand column shows the total rewards given to each of the network task classes, i.e. the inference task (blue), the forecasting task (cyan), and the reputer task (red), as well as the combined total in black. The top panel shows these rewards per time step and the bottom panel gives their cumulative sum over time. The other columns show the same information broken down into rewards for individual participants engaged in the inference (left-middle), forecasting (right-middle) and reputer (right) tasks.*

sybil, which would increase the number of participants while keeping their effective number approximately constant. As a result, the number ratio term $(N_{\text{eff}}/N)^{\beta}$ would decrease and help maintain constant entropy. Tests of this formulation show that $\beta = 0.25$ achieves this nearly exactly, and we adopt this as a fiducial value.

With the entropies in hand, we define the part of the total reward emitted per time step E_i that is allocated to each class as

$$U_i = \frac{(1-\chi)\gamma F_i E_i}{F_i + G_i + H_i},\ V_i = \frac{\chi\gamma G_i E_i}{F_i + G_i + H_i},\ W_i = \frac{H_i E_i}{F_i + G_i + H_i}. \tag{38}$$

In other words, each class of activity within a topic is allocated a reward proportional to the fraction of the total modified entropy $(F_i + G_i + H_i)$ that is generated by that class. The rewards for the inference and forecasting tasks each contain an additional factor ($(1-\chi)\gamma$ and $\chi\gamma$, respectively). These factors are included, because the reward split between the inference and forecasting tasks should additionally acknowledge the added value provided by the forecasting task. Fundamentally, the inference task is the engine of the Allora network; without any inferences to start with, there would be no network inference. By contrast, the network can function without the forecasting task, but is included to create context awareness and increase the accuracy of the network inference. Therefore, it is reasonable to modulate the reward allocation between both worker tasks according to the relative utility of the forecasting task.

The forecasting task utility χ is measured using a definition analogous to the one-out performance scores in Equation 16. We subtract the log-loss of the complete network inference $(\mathcal{L}_i)$ from that of the naive network $(\mathcal{L}_i^-)$, which is obtained by omitting all forecast-implied inferences:

$$T_i = \log \mathcal{L}_i^- - \log \mathcal{L}_i. \tag{39}$$

The performance score of the entire forecasting task T_i is positive if the removal of the forecasting task would increase the network loss, and is negative if its removal would decrease the network loss. We then apply a sigmoid function to the score and scale the result to let the forecasting task utility range from $\chi = (0.1, 0.5)$:

$$\chi = 0.1 + 0.4\sigma(aT_i - b), \tag{40}$$

where σ represents the sigmoid function and we adopt $a = 8$ and $b = 0.5$ to generate a relatively smooth transition around $T_i \approx 0.5$, with $\chi \approx 0.1$ for $T_i < 0$ and $\chi \approx 0.5$ for $T_i > 1$. We can now multiply the allocation of class rewards for the inference and forecasting tasks by $(1-\chi)$ and χ, respectively, and then renormalize with a factor γ to ensure that the total reward allocated to workers $(U_i + V_i)$ remains constant (otherwise, this would go at the expense of reputers). It is straightforward to demonstrate that this normalization factor reads

$$\gamma = \frac{F_i + G_i}{(1-\chi)F_i + \chi G_i}. \tag{41}$$

Substitution of Equation 41 into Equation 38 indeed results in $U_i + V_i = E_i(F_i + G_i)/(F_i + G_i + H_i)$, as desired.

Figure 3 illustrates the resulting incentive structure of the Allora network, using the same numerical experiment as considered in Figure 2. The best-performing participants (Worker 2 and Reputer 2) receive the largest share of the rewards

on average, allowing them to achieve the highest cumulative rewards at the end of the experiment. Toward the end of the experiment, the spread in rewards is larger for the reputers than for either of the two worker tasks, indicating that their adjusted entropy is lowest. As a result, reputers receive the smallest reward share. The fact that the inference task is allocated higher cumulative rewards than the forecasting task is mainly due to the low initial utility of the forecasting task, which is visible in Figure 2 as the steep decrease in the losses of the forecast-implied inferences during the first time steps. The forecasting task needs some time before it outperforms the individual inferences, which means that initially the inference task receives most of the worker reward share. This initial difference is partially overcome later on, due to the small spread in rewards per time step of the forecasting task and its correspondingly high entropy, which leads to a high reward allocation in comparison to the other tasks.

The above design represents a complete description of the differentiated incentive structure of a topic within the Allora network. The described rule set appropriately rewards workers for high-quality inferences obtained from their inference and forecasting tasks. It also rewards reputers according to their stake and consensus, allowing them to provide economic security to the topic. Finally, it incentivizes and rewards a high degree of decentralization, where a topic hosts a large number of participants that all make relevant contributions to network inferences and network security.

5 Conclusion

We have proposed Allora, a self-improving, decentralized machine intelligence network capable of translating a continuous data stream into a series of network inferences that outperform any individual inference existing within the network. Allora consists of worker nodes and reputer nodes, where the reputer nodes provide the economic security of the network by staking in the network and reporting on the accuracy of the worker nodes in reference to the ground truth. The worker nodes perform two different tasks. First, they provide inferences of the target variable under consideration (the 'inference task'). Second, they forecast the losses of the inferences of all worker nodes under the current conditions (the 'forecasting task'). This key ingredient correlates performance and context, and thereby makes the network context-aware. The network translates these forecasted losses into a single joint 'forecast-implied inference' per worker. The full set of original inferences and the forecast-implied inferences are then combined into a network inference through Allora's Inference Synthesis mechanism. We demonstrate that this network inference considerably outperforms the 'naive' network inference, obtained by excluding the forecasting task.

In addition to these functional developments, Allora also features a differentiated incentive structure, allowing network participants to be appropriately rewarded for specific behavior that aligns with the interests of the network. Workers are rewarded for high-quality inferences obtained from their inference and forecasting tasks, without any form of dilution by factors that might distract from their main purpose (cf. stake-weighted worker rewards). Reputers are rewarded according to their stake and consensus, implying that they act as the economic and functional guardians of the network. Finally, the network distributes rewards such that it maximizes the decentralization of the network, by rewarding groups of participants with high entropy. This setup alleviates possible attack vectors and contributes to the security and longevity of the network.

With these innovations, Allora addresses two major challenges in decentralized AI. First, Allora recognizes that different roles within the network require different incentive structures. Second, Allora acknowledges that selecting the best inference across a network of participants often depends on contextual details that themselves may require machine intelligence to be identified. By addressing these fundamental challenges in decentralized machine intelligence, the Allora network returns inferences that outperform the strongest network participant by definition, yet rewards each of them fairly for their contribution towards achieving this goal.

Allora's applications are without bounds, and its accessibility and transparency make state-of-the-art machine intelligence available to anyone. While the initial design focuses on supervised forms of AI, it will be natural to extend Allora's functionality to unsupervised AI and generative AI. With the versatility and accessibility of Allora, we foresee a future where machine intelligence will eventually become fully commoditized and integrated with the economy, technology, and society.

References

Banzhaf, J. 1965, Weighted voting doesn't work: A mathematical analysis, Rutgers Law Review, 19, 317–343

Buterin, V. 2014, Ethereum: A next-generation smart contract and decentralized application platform, https://github.com/ethereum/wiki/wiki/White-Paper, accessed: March 2024

Buterin, V. 2024, The promise and challenges of crypto + AI applications, https://vitalik.eth.limo/general/2024/01/30/cryptoai.html, accessed: March 2024

Bzdok, D., Nichols, T. E., & Smith, S. 2019, Towards Algorithmic Analytics for Large-scale Datasets, Nature Machine Intelligence, 1, 296–306

Craib, R., Bradway, G., Dunn, X., & Krug, J. 2017, Numeraire: A Cryptographic Token for Coordinating Machine Intelligence and Preventing Overfitting, https://numer.ai/whitepaper.pdf, accessed: March 2024

Fisher, R. A. 1922, On the Mathematical Foundations of Theoretical Statistics, Philosophical Transactions of the Royal Society of London Series A, 222, 309–368

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. 1991, Adaptive mixtures of local experts, Neural computation, 3, 79–87

Lightman, H., Kosaraju, V., Burda, Y., et al. 2023, Let's Verify Step by Step, CoRR, abs/2305.20050

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. 2017, in Proceedings of Machine Learning Research, Vol. 54, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, ed. A. Singh & J. Zhu (PMLR), 1273–1282

Nakamoto, S. 2008, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, accessed: March 2024

OpenAI. 2023, GPT-4 Technical Report, abs/2303.08774

Rao, Y., Steeves, J., Shaabana, A., Attevelt, D., & McAteer, M. 2021, BitTensor: A Peer-to-Peer Intelligence Market, abs/2003.03917

Shapley, L. S. 1953, A Value for n-Person Games, in Contributions to the Theory of Games II, ed. H. W. Kuhn & A. W. Tucker (Princeton: Princeton University Press), 307—317

Shazeer, N., Mirhoseini, A., Maziarz, K., et al. 2017, Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, CoRR, abs/1701.06538

Steeves, J., Shaabana, A., Hu, Y., et al. 2022, Incentivizing Intelligence: The Bittensor Approach, https://bittensor.com/academia, accessed: March 2024

Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, Attention is all you need, in Advances in Neural Information Processing Systems, 5998–6008