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*

Whitepaper v2024.8.14

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 (where lower losses indicate better performance). 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

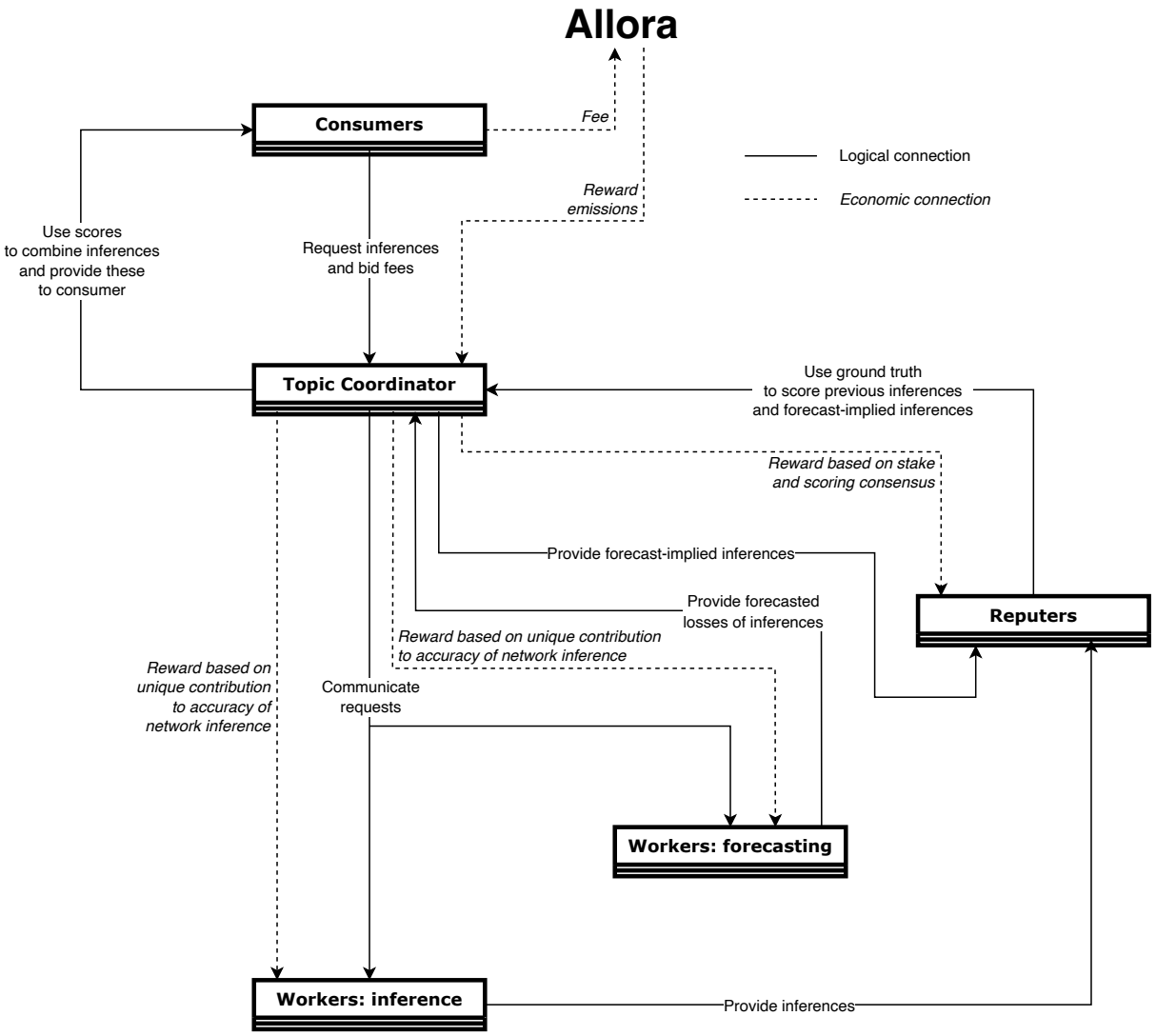

Figure 1: *Schematic representation of an Allora 'topic', which is a sub-network within the Allora ecosystem characterized by a specific AI target and loss function. Topics help organize the problems that Allora is solving, and they are used to coordinate the collaboration between network participants. 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.

3.1 The Allora Inference Synthesis Logic

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 logarithm of 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}:

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

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} and of the network loss $\mathcal{L}_{i-1}$ that was reported at the previous time step, which results in

$$R_{ijk} = \log \mathcal{L}_{i-1} - \log L_{ijk}. \tag{4}$$

Because lower losses indicate better performance, 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 through the derivative of a potential function $\phi_{p,c}(x)$:

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

where we define a simple potential function that is non-negative, convex, increasing, and twice smoothly differentiable:

$$\phi_{p,c}(x) = \ln\left[1 + \mathrm{e}^{p(x-c)}\right]. \tag{6}$$

This potential function is a smooth approximation of $\max(0, p(x-c))$ and has a derivative:

$$\phi'_{p,c}(x) = \frac{p}{\mathrm{e}^{-p(x-c)} + 1}, \tag{7}$$

where we adopt $p = 3$ and $c = 0.75$ as fiducial values. This functional form ensures that workers who provide inferences with negative expected regrets (i.e. $L_{ijk} > \mathcal{L}_{i-1}$) should be assigned low weights, with an effective scaling of $w_{ijk} \propto L_{ijk}^{-p}$, whereas positive contributions tend to a constant non-zero weight, with a transition point between both regimes at c. Before the forecasted regret is passed as the argument of the potential function's derivative, it is first normalized as

$$\hat{R}_{ijk} = \frac{R_{ijk}}{\sigma_j(R_{ijk}) + \epsilon}, \tag{8}$$

where σ_j indicates taking the standard deviation over all $j \in \{1, \ldots, N_\mathrm{i}\}$ and the second term in the denominator is added to avoid any division by zero. The constant ϵ (with default value $\epsilon = 0.01$) is set by the topic creator to set the numerical precision at which the network should distinguish differences in the logarithm of the loss. Normalization by the standard deviation of the forecasted regrets restricts $\hat{R}_{ijk}$ to values in relative proximity to zero. This normalization ensures that the network always obtains a non-zero weight for at least one inference (which without a normalization may not happen in case of all-negative forecasted regrets), while maintaining the intention of assigning extreme (i.e. near-zero or near-unity) weights to models with unusual forecasted regrets. Using these definitions, the network has access to a total

of $N_\text{i} + N_\text{f} \leq 2N_\text{w}$ 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]

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

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

with

$$\hat{\mathcal{R}}_{i-1,l} = \frac{\mathcal{R}_{i-1,l}}{\sigma_l(\mathcal{R}_{i-1,l}) + \epsilon}, \tag{11}$$

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:

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

[1] Recall that a worker may choose to engage (or not) in any of the tasks, so $2N_\text{w}$ 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.

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 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 logarithm of the 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 $\alpha \in (0, 1]$:

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

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.

3.2 Confidence Intervals on the Network Inference

In addition to providing network inferences I_i, the network also provides confidence intervals (CIs) on the network inference that reflect the spread of the inferences provided by the workers I_{il}. The CIs are based on percentiles P, which are chosen to be $P \in \{2.28, 15.87, 84.13, 97.72\}\%$ (with corresponding quantiles $q \equiv P/100$) to mimic the 1σ and 2σ limits of a Gaussian distribution, even if there is no guarantee that the distribution of worker inferences is Gaussian. When calculating the CIs, the weight w_{il} assigned to each worker inference is accounted for to ensure that inferences with low weight do not inflate the CIs.

Given q, the weighted quantile inference I_i^q is computed as follows. First, the worker inferences and their corresponding weights are sorted over dimension l such that $I_{il} \leq I_{i,l+1}$. For each sorted worker inference l, the cumulative sum of the weights is calculated as

$$c_{il} = \sum_{l'=1}^{l} w_{il'}. \tag{16}$$

These cumulative weights are then normalized to create a weighted cumulative distribution function (CDF):

$$C_{il} = \frac{c_{il} - 0.5 w_{il}}{\max_l c_{il}}, \tag{17}$$

where $\max_l c_{il}$ is the total sum of the weights. The term $0.5 w_{il}$ is subtracted to locate the quantile value at the middle of the weight associated with I_{il}. Finding the inference value I_i^q corresponding to the quantile q then requires a simple linear interpolation within the sorted list of data points using the weighted CDF. We define $l = \ell$ as the largest index where $C_{il} \leq q$. We then obtain I_i^q as

$$I_i^q = I_{i\ell} + \frac{q - C_{i\ell}}{C_{i,\ell+1} - C_{i\ell}} \left(I_{i,\ell+1} - I_{i\ell}\right). \tag{18}$$

This calculation of CIs accounts for the relative weight of each worker inference and only reflects the degree of internal disagreement among the workers providing inference to the network. It does not provide any insight into historical network performance in relation to the ground truth.

3.3 Numerical Simulation of Allora's Inference Synthesis and Performance

We now quantify the expected performance of the Allora architecture, obtained through a numerical simulation of the logic described in §3.1. The experiment is designed to closely mimic the live network performance without needing to run any actual inference models. Here we describe how the worker and reputer functionality is mimicked, and how the simulation is set up.

We consider a simple numerical experiment using mock data, where three workers generate inferences that predict the ground truth with some specified accuracy. The worker inference accuracy is characterized by an error (randomly drawn for each worker from a log-normal distribution with log-mean -0.700 and log-dispersion 0.176) and a bias (randomly drawn for each worker from a normal distribution with mean zero and dispersion 0.05). Each inference-providing worker is assumed to gain experience over time, which is modelled by multiplying its error and bias by a time-dependent factor $f_{\mathrm{xp},i}$, defined as

$$f_{\mathrm{xp},i} = \frac{1}{2}\left(1 + \mathrm{e}^{-0.03i}\right), \tag{19}$$

reflecting an exponential decay from unity to 0.5. At each time step, the error and bias of a different randomly-drawn worker is temporarily decreased by a factor 0.3 to model context-dependent outperformance. With these parameters, the inference provided by each worker during each time step is obtained by taking the ground truth and adding a difference drawn from a Gaussian distribution with mean equal to the worker's current inference bias and standard deviation equal to the worker's current inference error.

The workers also forecast each other's losses with different degrees of context-sensitivity, i.e. their ability to anticipate this temporary outperformance. In the simulation, we adopt a standard mean squared error (MSE) to described the loss. The worker forecasting accuracy is characterized by a logarithmic error (randomly drawn for each worker from a log-normal distribution with log-mean -0.222 and log-dispersion 0.176), a logarithmic bias (randomly drawn for each worker from a normal distribution with mean zero and dispersion 0.3), and a context sensitivity parameter f_{context} that is obtained by randomly drawing a number $x \in [0, 1]$ and applying a sigmoid function:

$$f_{\text{context}} = \sigma[a(x - b)], \tag{20}$$

to increase the incidence of values near zero and unity, with $a = 10$ and $b = 0.5$. As before, each forecast-providing worker is assumed to gain experience over time by multiplying the error and bias with $f_{\text{xp},i}$ from Equation 19. With these parameters, the forecasted losses provided by each worker during each time step are obtained by taking the logarithm of the actual loss for each worker's inference and adding a logarithmic difference drawn from a Gaussian distribution with mean equal to the worker's current forecasting bias and standard deviation equal to the worker's current forecasting error. We consider two such forecasted losses. The first forecasted loss is a perturbation of the actual loss including context-dependent outperformance, modelled as described above by randomly decreasing the error and bias of a worker each time step. The second forecasted loss is a perturbation of the loss corrected for any context-dependent outperformance, obtained by dividing the actual loss by a factor 0.3^2 (appropriate for the MSE loss adopted here). The final forecasted loss is obtained as a linear combination of the logarithms of these losses, where the logarithm of the actual loss is weighted by a factor f_{context}, the logarithm of the context-unaware loss is weighted by a factor $1 - f_{\text{context}}$, and the result is converted back into linear space. This accurately reflects the individual context awareness of each forecast-providing worker.

Finally, a set of five reputers is initialized by defining three parameters that describe the accuracy and definition of their reported losses. Each reputer is characterized by an error (randomly drawn for each reputer from a log-normal distribution with log-mean -1.000 and log-dispersion 0.097) and a bias (randomly drawn for each worker from a normal distribution with mean zero and dispersion 0.05). Additionally, we assume that reputers do not report the instantaneous loss, obtained by simply comparing each inference to the ground truth, but factor in a worker's historical performance by applying an exponential moving average to the instantaneous loss (analogously to e.g. the regret calculation of Equation 15), each adopting an individual value α_m (randomly drawn for each reputer from a log-normal distribution with log-mean -1.523 and log-dispersion 0.301). With these parameters, the losses reported by each reputer during each time step are obtained by taking the logarithm of the actual loss for each inference and adding a logarithmic difference drawn from a Gaussian distribution with mean equal to the reputer's bias and standard deviation equal to the reputer's error. The exponential moving average using each reputer's individual value of α_m is then applied to the logarithm of the losses, before converting these back to linear space.

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

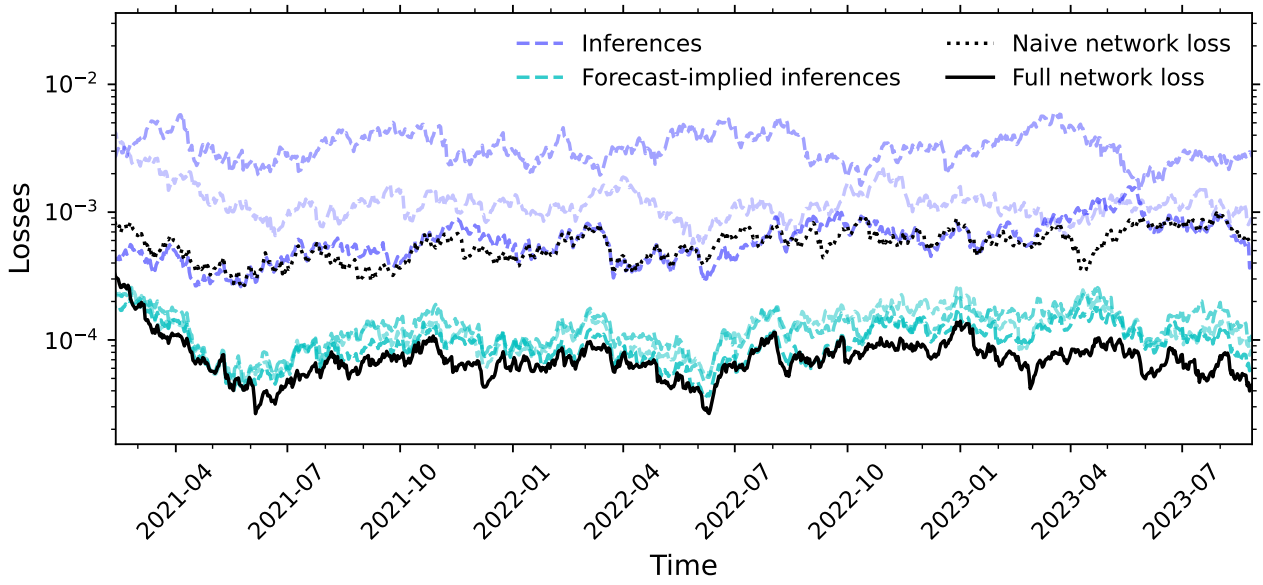

Figure 2: *Demonstration of Allora's self-improving intelligence and the accuracy improvement due to its context-aware Inference Synthesis mechanism. The dotted black line shows the naive network loss as a function of time, which is obtained by combining individual inferences (blue) without context awareness. 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.*

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 difference of the logarithm of the loss 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:

$$T_{ij} = \log \mathcal{L}^{-}_{ji} - \log \mathcal{L}_i, \tag{21}$$

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:

$$T_{ik} = (1 - f^{+})T^{-}_{ik} + f^{+}T^{+}_{ik}, \tag{22}$$

where we define the one-out score analogously to Equation 21:

$$T^{-}_{ik} = \log \mathcal{L}^{-}_{ki} - \log \mathcal{L}_i, \tag{23}$$

and the one-in score as

$$T^{+}_{ik} = \log \mathcal{L}^{-}_{i} - \log \mathcal{L}^{+}_{ki}. \tag{24}$$

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 22 is parameterized using f^+, which represents the fraction of permutations in a binomial experiment in which a worker appears solo:

$$f^+ = \frac{1}{2^{N_\mathrm{f}}}, \tag{25}$$

where N_f is the number of workers providing forecasted losses during the forecasting task.

The scores obtained in Equation 21 and Equation 22 facilitate the distribution of rewards to workers for inference and forecasting tasks. For a total reward allocated to each of these tasks per time step (U_i and V_i, both specified in §4.2), we use the scores to calculate the fraction of the rewards allocated to the two respective worker tasks that is received by each individual worker. For the inference task, these are defined as

$$u_{ij} = \frac{M(T_{ij})}{\sum_j M(T_{ij})}, \tag{26}$$

and for the forecasting task, these are

$$v_{ik} = \frac{M(T_{ik})}{\sum_k M(T_{ik})}. \tag{27}$$

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 satisfying these requirements is

$$M(T) = \phi_{p,c}\left[\frac{T}{\sigma(T) + \epsilon}\right], \tag{28}$$

where $\sigma(T)$ represents the standard deviation of all scores over the ΔN most recent time steps. The use of the potential function $\phi_{p,c}$ (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 second term in the denominator is added to avoid division by zero. The parameter p associated with the potential function controls the spread of the rewards. The fiducial parameter values are $p = 3$, $c = 0.75$, $\epsilon = 0.01$, 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

$$\hat{S}_{im} = \min\left(\frac{N_\mathrm{r} a_{im} S_{im}}{\sum_m a_{im} S_{im}}, 1\right), \tag{29}$$

where N_r is the number of reputers 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 returns unity for above-average stake fractions ($a_{im}S_{im}/\sum_m a_{im}S_{im} > 1/N_\mathrm{r}$), 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 unity at $a_{im}S_{im}/\sum_m a_{im}S_{im} = 1/N_\mathrm{r}$. 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

$$\mathbf{L}_{im} = \{\mathcal{L}_{im}, \mathcal{L}_{ijm}, \mathcal{L}_{ikm}, \mathcal{L}_{im}^-, \mathcal{L}_{lim}^-, \mathcal{L}_{kim}^+\}, \tag{30}$$

and define the consensus loss vector as

$$\log \mathbf{L}_i = \frac{\sum_m \hat{S}_{im} \log \mathbf{L}_{im}}{\sum_m \hat{S}_{im}}. \tag{31}$$

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

$$T_{im} = \left[\frac{\|\log(\mathbf{L}_{im}/\mathbf{L}_i)\|}{\|\log \mathbf{L}_i\|} + \epsilon_\mathrm{r}\right]^{-1}, \tag{32}$$

where the first term expresses the relative proximity and $\epsilon_\mathrm{r} = 0.01$ 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:

$$T_i = \frac{\sum_m S_{im} T_{im}}{\sum_m S_{im}}. \tag{33}$$

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

$$a_{im} \to a_{im} + \lambda \frac{\mathrm{d} \ln T_i}{\mathrm{d} a_{im}}, \tag{34}$$

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 34 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 34 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.

For a total reward allocated to reputers per time step (W_i, specified in §4.2), we now calculate the fraction of the rewards allocated to reputers that is received by each individual reputer as

$$w_{im} = \frac{(S_{im} T_{im})^p}{\sum_m (S_{im} T_{im})^p}, \tag{35}$$

where the multiplication by stake and consensus score ensures the appropriate dependence of the reward fraction 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:

$$S_{i+1,m} = S_{im} + w_{im} W_i, \tag{36}$$

where W_i is the total reward allocated to reputers (see §4.2). 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 to a topic t in a given time step $E_{t,i}$ 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:

$$F_i = -\sum_j f_{ij} \ln(f_{ij}) \left(\frac{N_{\mathrm{i,eff}}}{N_\mathrm{i}} \right)^\beta, \; G_i = -\sum_k f_{ik} \ln(f_{ik}) \left(\frac{N_{\mathrm{f,eff}}}{N_\mathrm{f}} \right)^\beta, \; H_i = -\sum_m f_{im} \ln(f_{im}) \left(\frac{N_{\mathrm{r,eff}}}{N_\mathrm{r}} \right)^\beta, \tag{37}$$

where we have defined modified reward fractions per class as

$$f_{ij} = \frac{\tilde{u}_{ij}}{\sum_j \tilde{u}_{ij}}, \; f_{ik} = \frac{\tilde{v}_{ik}}{\sum_k \tilde{v}_{ik}}, \; f_{im} = \frac{\tilde{w}_{im}}{\sum_m \tilde{w}_{im}}. \tag{38}$$

Here, the tilde over the reward fractions indicates that we have applied an exponential moving average:

$$\tilde{u}_{ij} = \alpha u_{ij} + (1-\alpha) \tilde{u}_{i-1,j}, \; \tilde{v}_{ik} = \alpha v_{ik} + (1-\alpha) \tilde{v}_{i-1,k}, \; \tilde{w}_{im} = \alpha w_{im} + (1-\alpha) \tilde{w}_{i-1,m}, \tag{39}$$

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 37, because the increase of the actual entropy with $\ln N$ might otherwise incentivize a sybil attack, in which any class of topic participants 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 37 has been added to prevent this. It contains an effective number of participants, which is defined as

$$N_{\mathrm{i,eff}} = \frac{1}{\sum_j f_{ij}^2},\ N_{\mathrm{f,eff}} = \frac{1}{\sum_k f_{ik}^2},\ N_{\mathrm{r,eff}} = \frac{1}{\sum_m f_{im}^2}. \tag{40}$$

For a uniform reward distribution, $N_{\mathrm{eff}} = N$ by definition. For strongly unequal reward distributions, $N_{\mathrm{eff}} \ll N$. With the addition of the $(N_{\mathrm{eff}}/N)^\beta$ term in Equation 37, any sybil attack would necessarily dilute the individual rewards of the sybil, which would increase the number of participants while keeping their effective number approximately constant. As a result, the number ratio term $(N_{\mathrm{eff}}/N)^\beta$ 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.

Given a total reward emitted to a topic t within a time step $E_{t,i}$, we use the entropies to define the part of the rewards that is allocated to each class as

$$U_i = \frac{(1-\chi)\gamma F_i E_{t,i}}{F_i + G_i + H_i},\ V_i = \frac{\chi\gamma G_i E_{t,i}}{F_i + G_i + H_i},\ W_i = \frac{H_i E_{t,i}}{F_i + G_i + H_i}. \tag{41}$$

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 21. We subtract the logarithm of the 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:

$$T_i = \log \mathcal{L}_i^- - \log \mathcal{L}_i. \tag{42}$$

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 define a modified ratio between the forecasting task performance score and the sum of all inference scores as

$$\tau_i \equiv \alpha \frac{T_i - \min\left(0, \max_j T_{ij}\right)}{\left|\max_j T_{ij}\right|} + (1-\alpha)\tau_{i-1}, \tag{43}$$

with $\alpha = 0.1$, and apply a piecewise linear transformation to set the forecasting task utility within the range $\chi = (0.1, 0.5)$:

$$\chi = \begin{cases} 0.1 & \text{if } \tau_i < 0, \\ 0.4\tau_i + 0.1 & \text{if } 0 \le \tau_i < 1, \\ 0.5 & \text{if } \tau_i \ge 1. \end{cases} \tag{44}$$

This represents a linear transition from $\chi = 0.1$ to $\chi = 0.5$ in the range $\tau_i = [0, 1]$. 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

$$\gamma = \frac{F_i + G_i}{(1-\chi)F_i + \chi G_i}. \tag{45}$$

Substitution of Equation 45 into Equation 41 indeed results in $U_i + V_i = E_{t,i}(F_i + G_i)/(F_i + G_i + H_i)$, as desired. The rewards per class specified in Equation 41 can now be combined with the reward fractions from Equation 26, Equation 27, and Equation 35 to obtain the absolute rewards paid out to individual network participants:

$$U_{ij} = u_{ij}U_i,\ V_{ik} = v_{ij}V_i,\ W_{im} = w_{ij}W_i. \tag{46}$$

Figure 3 illustrates the resulting incentive structure of the Allora network, using the same numerical experiment as considered in Figure 2 and additionally adopting a stochastic model to describe the total reward emission, which is the main boundary condition needed to simulate the reward distribution (black line in the left panels of Figure 3). 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.

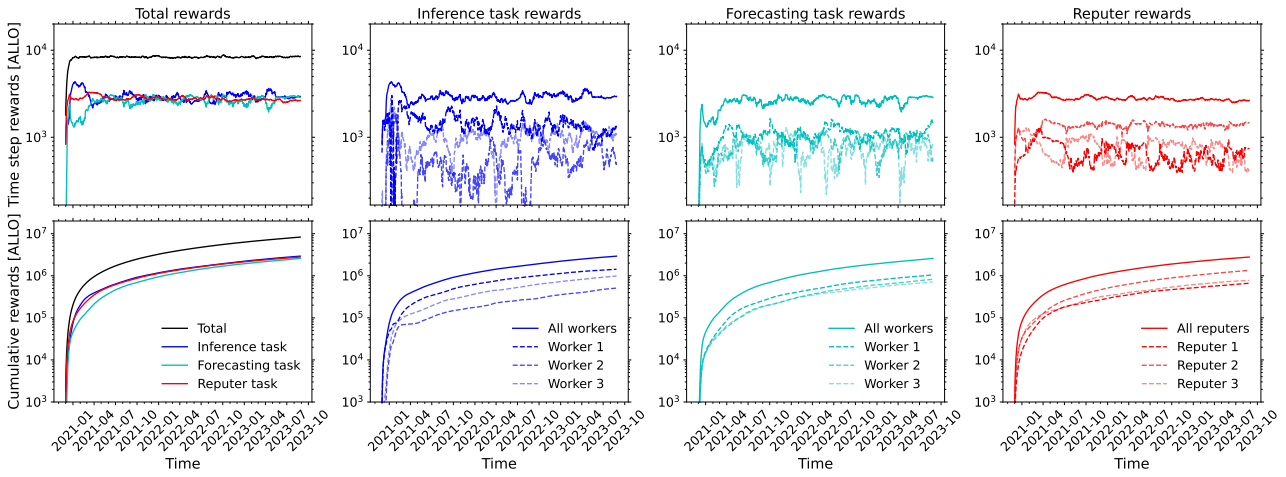

Figure 3: *Demonstration of the incentive structure of the Allora network, showing how rewards are distributed among different network participants over time. The left-hand column shows the total rewards given to each of the network task classes, i.e. the inference task (blue), the forecasting task (cyan), 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.*

As a result, reputers receive the smallest reward share. The allocation of higher cumulative rewards to the inference task 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 raw inferences, which means that initially the inference task receives most of the worker reward share. This initial difference is partially overcome later on by the small spread in rewards per time step of the forecasting task and its correspondingly high entropy, which leads to a high reward allocation relative 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 Allora Tokenomics

5.1 Token Emission Rate

The Allora token (ALLO) is minted by the Allora network to facilitate the exchange of value by network participants. Specifically, the token holders can choose to engage in the following forms of token utility:

1. The ALLO token can be used to purchase inferences that are generated by the network. Allora adopts a pay-what-you-want (PWYW) model, where token holders choose the ALLO fee they are paying for an inference (see §5.3 for details).

2. The ALLO token can be used to pay the access fee for topic creation, or for participating in the network as a worker, reputer, or network validator. The access fee price is variable.

3. The ALLO token can be used by reputers and network validators to stake, and by any other token holders to delegate stake to a reputer or network validator. Staking reputers, staking network validators, and delegating token holders receive rewards in ALLO.

4. The ALLO token is used by the network to pay out rewards to network participants. For workers, these rewards are proportional to their unique contribution to the network accuracy. For reputers and network validators, these rewards are proportional to their stake and the consensus between their output and those of other participants (see §4). The total amount of ALLO emitted by the network is offset by any fees collected, implying that the ALLO in circulation can increase (corresponding to inflation) or decrease (corresponding to deflation) depending on the market dynamics.

The above comprehensively describes the utility of the ALLO token. ALLO tokens will not in any way represent any shareholding, participation, right, title, or interest in any company, enterprise or undertaking, nor will the tokens entitle token holders to any promise of dividends, revenue, capital, profits or investment returns.

Token emissions serve the purpose of providing liquidity for reward payouts. The general philosophy adopted by the Allora tokenomics revolves around two key goals. First, ALLO emissions are subjected to a smoothened form of a Bitcoin-like schedule (Nakamoto, 2008) to maintain long-term rewards in a limited-supply economy. Second, the annual percentage yield (APY) earned per staked token should be stable around major token unlocks to disincentivize these tokens from being dumped on the market. Together, these goals imply a simple emission design that is described as follows.

The total emission at time step i is given by

$$\mathcal{E}_i = e_i \frac{\mathcal{N}_{\text{staked},i}}{N_{\text{epochs}}}, \tag{47}$$

where $\mathcal{E}_i$ is the total number of tokens emitted as rewards, e_i is the monthly emission per unit staked token, $\mathcal{N}_{\text{staked},i}$ represents the number of staked tokens, and N_{epochs} is the monthly number of time steps (epochs) during which rewards are paid out. We define a target monthly emission per unit staked token that the network strives for:

$$\hat{e}_{\text{target},i} = \frac{f_e \mathcal{T}_{\text{total},i}}{\mathcal{N}_{\text{staked},i}} \frac{\mathcal{N}_{\text{circ},i}}{\mathcal{N}_{\text{total},i}}, \tag{48}$$

where $\mathcal{T}_{\text{total},i}$ is the total number of tokens held by the network treasury, $\mathcal{N}_{\text{total},i} = 1,000,000,000$ is the total token supply, $\mathcal{N}_{\text{circ},i}$ is the circulating supply, and $\mathcal{N}_{\text{staked},i}$ is the staked supply. The factor $f_e = 0.025\ \text{month}^{-1}$ represents the fraction of the network treasury that would ideally be emitted per unit time. Equation 48 states that the total emission per unit time should be equal to $f_e \mathcal{E}_{\text{total},i}$ in the limit where the entire token supply would be in circulation.

If only a small fraction of the circulating supply is staked (i.e. $\mathcal{N}_{\text{staked},i} \ll \mathcal{N}_{\text{circ},i}$), then Equation 48 can result in an undesirably high APY. Allora therefore limits the APY to 12% and correspondingly limits the monthly percentage yield (MPY) to 0.95%. This is achieved by limiting the target emission per unit staked token. The MPY depends on the monthly emission per unit staked token as

$$\xi_i = f_{\text{stakers},i} e_i, \tag{49}$$

where $f_{\text{stakers},i}$ represents the fraction of the total token emission during the previous epoch that was paid to reward staking network participants, i.e. reputers and network validators. The maximum monthly emission per unit staked token given a maximum MPY ($\xi_{\text{max}} = 0.0095$) then follows as

$$\hat{e}_{\text{max},i} = \frac{\xi_{\text{max}}}{f_{\text{stakers},i}}, \tag{50}$$

so that the target monthly emission per unit staked token becomes

$$\hat{e}_i = \min\left(\hat{e}_{\text{target},i}, \hat{e}_{\text{max},i}\right). \tag{51}$$

While Equation 48-Equation 51 set the target emission per unit staked token, in practice this expression introduces undesired behavior around major token unlocks. The jumps in $\mathcal{N}_{\text{staked}}$ and $\mathcal{N}_{\text{circ}}$ may result in APY drops and would thus incentivize holders to add to the selling pressure normally associated with token unlocks. To decrease the selling pressure and maintain a healthy economy, Allora adjusts the token emission to stabilize the emission per unit staked token. This emission smoothing is achieved by applying an exponential moving average to the monthly emission per unit staked token:

$$e_i = \alpha_e \hat{e}_i + (1 - \alpha_e) e_{i-1}, \tag{52}$$

where $\alpha_e = 0.1$ is a parameter setting the degree of smoothing when the update cadence is one month. It can easily be transformed to other cadences using the conversion:

$$\hat{\alpha}_e = 1 - (1 - \alpha_e)^{\Delta t/\text{month}}, \tag{53}$$

where $\hat{\alpha}_e$ is the recalibrated form of α_e appropriate for an update time step Δt.

The above framework defines the token emission in the absence of any token supply flowing back into the network treasury. In practice, all fees collected by the network are added to the network treasury, where they are used to pay out reward emissions first, before any additional tokens are minted. This means that, in practice, the network treasury will drain more slowly than the naive exponential decay of $f_e = 0.025\ \text{month}^{-1}$, as network usage generates fees that sustain its economy long-term and maintain a high APY for token stakers.

Figure 4 quantitatively illustrates how the APY depends on the mechanisms discussed above, assuming that holders stake approximately 60% of the circulating token supply. For illustration, we assume a scenario where a major token unlock takes place after one year. Without the emission smoothing of Equation 48-Equation 53, the APY experiences a sharp drop around this unlock (dotted line). Including emission smoothing remedies this jump, resulting in a smooth evolution of the APY that incentivizes token holders to continue staking their tokens even around major unlocks (dashed line). Finally, the addition of fee revenue (here assumed at a level of 2% of the unstaked circulating supply per month) results in a stable APY long-term. The APY remains high, because fee revenue is used to pay out rewards before minting new tokens.

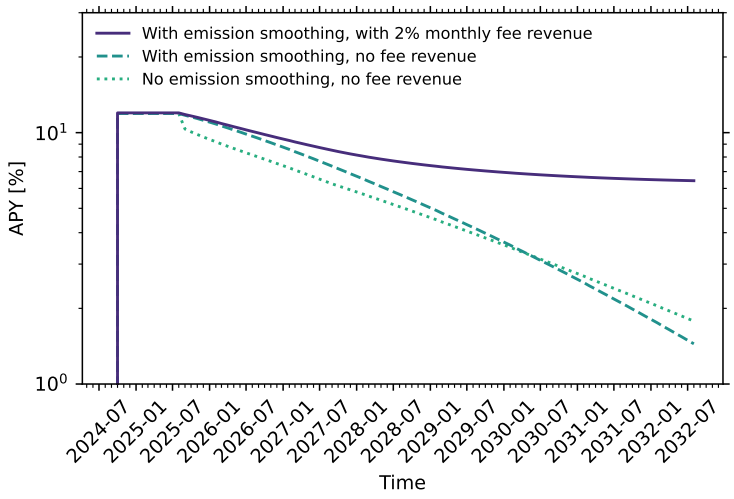

Figure 4: *Illustration of the APY earned by staking ALLO holders under the Allora Network tokenomics design. The figure shows the predicted APY evolution for three different setups, illustrating that the addition of emission smoothing (from Equation 48-Equation 53) and fee revenues result in a long-term sustainable APY.*

5.2 Reward Distribution to Topics and Network Validators

Given the total emission $\mathcal{E}_i$ at time step i, we now specify how the emission is distributed over network validators (who secure network communications) and topics (where the emission is used to pay out rewards as described in §4). The emission allocated to reward an individual network validator v is given by

$$E_{v,i} = f_{\mathrm{v}} \mathcal{E}_i \frac{S_{v,i}}{\sum_v S_{v,i}}, \tag{54}$$

where $f_{\mathrm{v}} = 0.25$ represents the fraction of the total reward emission $\mathcal{E}_i$ that is allocated to network validators, and $S_{v,i}$ is the stake of network validator v at time step i. Conversely, the emission allocated to pay out rewards to participants within a topic t is given by

$$E_{t,i} = (1 - f_v) \mathcal{E}_i \frac{w_{t,i}}{\sum_t w_{t,i}}, \tag{55}$$

where the factor $1 - f_v$ indicates that we are considering the complement of the emission allocated to network validators, and $w_{t,i}$ represents the topic reward weight, analogous to the validator stake $S_{v,i}$ in Equation 54. The topic reward weights are defined using an exponential moving average:

$$w_{t,i} = \alpha_t \hat{w}_{t,i} + (1 - \alpha_t) w_{t,i-1}, \tag{56}$$

where $\hat{w}_{t,i}$ represents the target weight. We adopt $\alpha_t = 0.5$ on a weekly timescale (corresponding to $\hat{\alpha}_t = 0.9375$ if the update cadence were monthly). The target weight is defined as a power law of the total amount staked by reputers within a topic and the effective fee revenue from the topic:

$$\hat{w}_{t,i} = S_{t,i}^{\mu} C_{t,i}^{\nu}. \tag{57}$$

Here, $S_{t,i}$ represents the total stake of reputers within topic t and $C_{t,i}$ represents the effective fee revenue collected by topic t, summing all fees paid by consumers utilizing the network and access fees paid by workers and reputers. The quantity $C_{t,i}$ decays each epoch by subtracting an amount $\Delta C_{t,i} = N_{\mathrm{epochs,w}}^{-1} C_{t,i}$ (where $N_{\mathrm{epochs,w}}$ is the number of epochs per week) and thus captures recent revenue. The exponents μ and ν indicate the relative importance of either quantity; we adopt equal weights of $\mu = 0.5$ and $\nu = 0.5$, implying that Equation 57 corresponds to a geometric mean.

5.3 Consumer Fee Pricing Model

The fee revenue $C_{t,i}$ is the accumulation of all fees paid by consumers into topic t over the preceeding week. These fees are set according to a pay-what-you-want (PWYW) model. PWYW models are free-form methods for facilitating price discovery, that may be accompanied by a minimum price, a suggested price, or no price floor at all. Commodities suitable for PWYW pricing are characterized by (Raju & Zhang, 2010): a low marginal cost, a fair-minded customer, a product that can be sold credibly at a wide range of prices, a strong relationship between buyer and seller, and a very competitive market place. Most, if not all, of these conditions apply to the Allora ecosystem in some capacity. In some cases, a PWYW model can result in a higher fee revenue than when imposing a fixed price, with the fee payments being Pareto-distributed on average (e.g. Chao et al., 2015). Non-zero payments are generally incentivized by information transparency,

e.g. through social conformity by communicating the price paid by other consumers, or through a common interest by considering the cost price of the service provider.

Within the Allora Network, the guaranteed token emission structure may seem to disincentivize non-zero payments. After all, the emission means that network participants receive rewards even in the absence of any fee revenue, thereby disrupting the feedback loop that would otherwise stimulate price discovery. However, the competition between topics overcomes this apparent disconnect and reintroduces the feedback loop between fees and rewards, because the topic weight in Equation 57 is proportional to the topic's accumulated consumer fees. If consumers within a topic pay zero fees, then the topic weight tends to zero, the participants within that topic receive no rewards, and the token emission will be distributed over the other topics. This system may only be taken advantage of by forming a cabal that enforces paying zero fees across all topics. However, such a situation would allow any fee payment to a topic to instantly boost that topic's weight greatly relative to all other topics, resulting in a negligibly small cost for any topic to attract all rewards. As such, the Allora PWYW model creates a healthy competition between topics to attract fee revenue, which naturally drives price discovery across the network.

6 Discussion

6.1 Limitations and Future Work

Allora achieves a form of self-improving, decentralized machine intelligence that fundamentally supports the combination of models of any nature. The description provided in this paper is specifically suited for supervised regression tasks, where inferences are provided for numeric target variables, and a ground truth is available to evaluate these inferences. However, Allora is suitable for extension to other forms of AI, including supervised classification, unsupervised learning or clustering, and generative AI.

Adapting Problems to Allora's Design. The extension of Allora's utility to other forms of AI can be achieved by changing the network design, but instead a problem set can often also be transformed to fit Allora's native supervised regression framework. This transformation typically involves encoding categorical data in numerical formats or defining new tasks that can be represented as continuous output. For classification tasks, it is often feasible to infer probabilities and compare these with a probabilistic ground truth later. Alternatively, some categorical data are ordinal, i.e. they have a meaningful order, which allows them to be encoded on a continuous numerical scale. For such problems, inferences can take on any real value that is subsequently rounded or binned back to the ordinal categories. Similar transformations exist for unsupervised clustering tasks, where domain-informed clusters may be used to define a membership probability as the target variable (thus transforming the problem into a supervised classification task). Alternatively, a clustering problem can sometimes be redefined as a dimensionality reduction problem where the original data are reconstructed using an autoencoder, and the reconstruction error acts as the loss function for a regression-based dimensionality reduction task. More broadly, any of the above problems may be addressed by using embedding techniques to transform categorical variables into a continuous vector space. This could naturally extend Allora's applicability to natural language processing and generative language models. Finally, Allora's modular topic structure enables setting up multiple topics to collaborate in an adversarial configuration, fostering a dynamic learning environment where different modules can iteratively improve through competition or cooperative adversarial tactics.

Adapting Allora to Other AI Forms. Despite Allora's innate flexibility, there exist problems that are unsuitable for the current, regression-oriented design of the network. If we consider the specific example of traditional classification problems, full compatibility with Allora would require suitable adjustments of:

1. the forecasted loss in Equation 2 to match the appropriate objective function;
2. the weighted averages in Equation 3 and Equation 9 to be meaningful for discrete, unordered variables;
3. the regrets of Equation 4 and Equation 15 to measure the change in classification effectiveness.

Subsequent adjustment of the potential function of Equation 6 (used to set weights in Equation 5 and Equation 10) and the regret normalizations of Equation 8 and Equation 11 may improve Allora's performance in classification tasks. Such optimizations will help Allora expand its utility and match the high standard that it already reaches in regression tasks.

A natural part of extending Allora to other forms of AI involves investigating its behavior across a variety of loss (or objective) functions. Fundamentally, Allora supports the use of any such function, but adaptations of the network design should be informed by the concrete properties of these objectives. For classification problems, relevant loss functions include hinge loss and binary or categorical cross-entropy loss, which all rely on predicted probabilities of a label occurring. Alternatively, it is possible to use incidence-based performance metrics, such as the accuracy, precision, recall, or F1 score, all of which may be inverted to satisfy the standard that lower values are better. For unsupervised learning problems, common objective functions include reconstruction loss, clustering loss, the Kullback & Leibler (1951) divergence, or the Silhouette Score (Rousseeuw, 1987), which all represent metrics to quantify the model's ability to identify similarities in data. For generative AI problems, common objective functions include adversarial loss, feature matching loss,

the Inception Score (Salimans et al., 2016), or the Fréchet Inception Distance (Heusel et al., 2017), which all represent metrics to quantify the similarity between generated data and a reference data set.

Future Research Directions. Allora's extension to various AI domains also necessitates further applied research, focused on identifying and developing concrete use cases, specifically those that capitalize on Allora's unique context awareness. This research should explore sectors where dynamic data environments are prevalent, such as financial markets, healthcare, weather, logistics, or real-time predictive maintenance systems. Concrete examples are discussed in §6.2. Applications in these areas will not only demonstrate Allora's practical utility, but may also increase its ability to handle real-world, dynamic information.

State-of-the-Art Forecaster Models. To maximize Allora's context-awareness provided by the forecasting task, we require state-of-the-art forecaster models and data streams that are tailored to the network's needs. This involves creating predictive models that accurately interpret and contextualize the historical performance of workers. Specific examples are machine learning models that incorporate temporal dynamics, such as recurrent neural networks with long short-term memory (Hochreiter & Schmidhuber, 1997) or gated recurrent units (Cho et al., 2014), which could considerably improve the network's accuracy by better representing sequences or time-series data. While part of the feature data used by these models will be sourced by the workers themselves, Allora will additionally provide a curated data stream supporting the forecasting task, including e.g. the worker inference losses, the worker inference scores, the forecasted losses from (and for) all workers, the rewards paid out to workers for the inference task, the total rewards paid out to the inference and forecasting tasks across all workers, and the network inference loss. These data will help ensure that the workers providing loss forecasts have access to all necessary information that may forecast worker performance, allowing Allora to optimize its context-aware inferences, and ultimately leading to more reliable and actionable machine intelligence.

Optimization and Scalability. As Allora grows in terms of both capability and scale, the network's architecture will require continuous optimization and development. This includes enhancing the scalability of the network to handle larger numbers of participants without any loss in accuracy or performance, as well as providing support for cross-topic connectivity. There exists a plethora of unexplored use cases that leverage a combination of multiple topics within Allora (e.g. setting up adversarial configurations), and the network will need continued development to fully realize its full potential in these emergent areas. In addition, ongoing refinement of the network's incentive structures is essential to ensure that all participants are correctly incentivized to contribute high-quality data and inferences. This will involve continued optimization of the reward mechanisms while minimizing potential attack vectors and maximizing long-term network health. Furthermore, as Allora expands into new AI domains, it will be necessary to periodically revisit and possibly redesign certain aspects of its architecture to ensure optimal performance across diverse applications. There exists a natural tension between optimizing the general applicability of Allora and maximizing the network performance in individual applications. This tension will continue to drive development and improvement to sustain Allora's position at the frontier of decentralized machine intelligence.

6.2 Allora Unlocks a World Shaped by Collective Intelligence

Allora's decentralized and context-aware machine intelligence has the potential to transform various aspects of society. By harnessing the collective intelligence of a diverse network of participants, Allora transcends traditional AI applications and unlocks innovative solutions across various sectors. We identify an initial, non-exhaustive set of six key areas where Allora's unique capabilities can drive significant progress: improved decision-making, democratization of AI, economic and social impact, privacy and security, innovation and collaboration, and sustainability and efficiency. These use cases demonstrate how Allora's approach not only democratizes access to cutting-edge AI, but also fosters a collaborative environment that helps maximize the utility and impact of machine intelligence.

Improved Decision-Making. Allora allows individuals and businesses of any size to access and leverage collective intelligence from a variety of sources. Contrary to centralized AI models, which are often biased towards singular data sets, Allora's context-aware design ensures that the network always relies on the most relevant and accurate insights, tailored to specific local conditions and needs across various sectors, including e.g. small businesses, financial markets, and healthcare. Small businesses can use context-aware AI-driven insights to optimize operations and customer engagement for their specific situation without requiring access to large-scale resources. In financial markets, decentralized AI can analyze real-time data from various sources with a variety of models to provide traders and investors with precise, context-aware insights, providing more accurate forecasts, better investment strategies, and improved risk management. In healthcare, Allora's context awareness will help identify bespoke solutions that are the best suited for individual patients.

Open Democratization of AI. Allora decentralization overcomes the siloed nature of traditional AI solutions. This enables individuals and small entities to contribute data and algorithms and receive precise, context-aware insights and monetary rewards in return. This open participation structure means that even those without significant resources become stakeholders in the network and gain ownership of the collective intelligence provided by Allora. It represents a general equalizer that increases e.g. market efficiency, the rate of innovation and development, and the agency of small communities. Allora heralds a future wherein machine intelligence belongs to everyone.

Wide Economic and Social Impact. Allora's decentralized infrastructure naturally provides economic and social benefits to community-driven projects. For instance, small financial cooperatives can leverage decentralized AI to better serve their members with tailored financial products and services. Retail investors and traders gain access to the state-of-

the-art in price prediction, fair market value forecasting, and risk assessment, thereby enhancing their decision-making capabilities and contributing to overall market efficiency. A similar impact is expected in real-world settings. Local agricultural communities can combine their data to get precise, context-aware recommendations on crop management, livestock, and weather, improving yields and sustainability. Hospitals and small doctor's offices will be able to contribute decentralized and anonymized data to Allora to create tailored and personalized treatments. In general, Allora's machine intelligence will provide small communities with insights and a degree of agency that would otherwise be restricted to well-funded monoliths.

Enhanced Privacy and Security. The decentralized nature of the Allora network inherently enhances data privacy and security. The full volume of data underpinning Allora's network inference is never concentrated in a single repository, reducing the risk of breaches. Workers only provide inferences without disclosing the underlying model or data set. This ensures that Allora's context-aware AI respects data and model sovereignty, leaving all relevant information under the control of the original contributors. This approach fundamentally enhances trust and security compared to traditional monolithic forms of AI, unlocking the application of machine intelligence in privacy-sensitive areas such as healthcare, legal proceedings, research and development, smart appliances, and financial sectors such as fraud detection and account monitoring.

Accelerated Innovation and Collaboration. Allora facilitates direct peer-to-peer collaboration. Researchers, developers, and small businesses can work together without intermediaries, leveraging the network's context-aware Inference Synthesis mechanism to build on each other's work. This leads to innovative solutions, incorporating all specialist domain expertise that is relevant to the current conditions. This enables effective collaboration and exchange of insights within small, localized communities, while fostering the creation of global networks of domain experts who can leverage their unique joint intelligence. Allora flattens barriers to entry, and its inherently collaborative infrastructure stimulates rapid and efficient innovation. Its decentralized model accelerates the development of new technologies and solutions, driving collaborative progress across an unbounded range of applications, including e.g. medicine, healthcare, finance, environmental science, agriculture, logistics, and infrastructure.

Ubiquitous Sustainability and Efficiency. Allora has the potential to play a unique role in the development of localized production and distribution of food and energy, as well as waste management. Community-led renewable energy initiatives can use Allora's context-aware AI to optimize their infrastructure and its operation. Similar applications exist in managing localized food production and small agricultural initiatives, which are naturally more easily disrupted by unexpected events or changing conditions. Decentralized waste management can be adjusted using real-time data from various local sources, making the process more efficient and environmentally friendly. These applications all benefit from Allora's ability to source insights globally and apply them optimally to the local conditions. As a result, Allora allows bottom-up, sustainable solutions to become a reality.

In summary, Allora's decentralized and context-sensitive machine intelligence demonstrates the power of integrating diverse data and algorithms from many participants, resulting in more precise and contextually relevant decision-making. This open and collaborative approach allows individuals and small entities to contribute and benefit simultaneously. The decentralized structure ensures data privacy and security, while stimulating innovation through direct peer-to-peer collaboration. A world shaped by Allora's collective intelligence leverages the context-based relative importance of diverse insights for better decision-making, making advanced AI accessible to all, maintaining data privacy through decentralization, and encouraging innovation through collaboration. Allora not only impacts complex problem solving, but it also creates a future where the advantages of AI can be shared and utilized by anyone.

7 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: August 2024

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

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

Chao, Y., Fernandez, J., & Nahata, B. 2015, Journal of Behavioral and Experimental Economics, 57, 176

Cho, K., van Merrienboer, B., Gulcehre, C., et al. 2014, arXiv e-prints, arXiv:1406.1078

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: August 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

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. 2017, arXiv e-prints, arXiv:1706.08500

Hochreiter, S. & Schmidhuber, J. 1997, Neural Computation, 9, 1735

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

Kullback, S. & Leibler, R. A. 1951, The Annals of Mathematical Statistics, 22, 79

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: August 2024

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

Raju, J. & Zhang, Z. 2010, Smart Pricing: How Google, Priceline, and Leading Businesses Use Pricing Innovation for Profitability (Pearson Education)

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

Rousseeuw, P. J. 1987, Journal of Computational and Applied Mathematics, 20, 53

Salimans, T., Goodfellow, I., Zaremba, W., et al. 2016, arXiv e-prints, arXiv:1606.03498

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: August 2024

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