Computer Communications 251 (2026) 108467

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Cognitive oracles: On-chain explainable machine learning☆

Marco Esposito *, Francesco Bruschi, Donatella Sciuto

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

ARTICLE INFO

Dataset link: https://github.com/okrame/xAI-oracle

Keywords:
Blockchain
Oracle
Machine learning
Snarks

ABSTRACT

Blockchains offer a mechanism for executing code in a credible, transparent and uncensorable way. These characteristics lend themselves to the implementation of applications that guarantee different actors regarding the credible execution of mechanisms, including insurance and impartial certification. A significant challenge in this domain is enabling smart contracts to reliably access real-world state information. We propose a "cognitive" oracle architecture that makes complex and ambiguous conditions — such as identifying specific elements within an image — verifiable on-chain. Unlike traditional oracles that focus on scalar data feeds, our oracle integrates machine learning classifiers with SNARK proofs to provide trustworthy semantic evaluations. To mitigate inaccuracies and faults, the oracle also embeds a jury-based dispute resolution layer supported by verifiable machine learning explanations. We evaluate this oracle system in the context of a carbon credit application based on small-size neural network inferences. The results and cost analyses indicate that the economic viability of such an oracle depends on minimizing dispute frequencies and selecting optimal configurations. The inclusion of explanation methods adds a layer of game-theoretic complexity but also enhances the quality of the jury's work, as well as overall transparency and trust.

1. Introduction

The increasing adoption of blockchain technologies significantly influences [1] the potential of digital processes. Distributed Applications, also known as dApps, represent a significant paradigm change, particularly with respect to the execution guarantees and the neutral credibility that they offer. While traditional applications utilize centralized servers to manage data and execute required programs and processes, a dApp employs blockchain as its foundational infrastructure, ensuring transparent, deterministic, unstoppable and uncensorable execution. They are founded upon the usage of self-executing computer programs or transaction protocols known as smart contracts [2], which are utilized to automate procedures and transactions in a reliable and credible way. Such publicly verifiable smart contracts reduce the need for intermediaries, providing added automation and security for on-chain executable code across various domains, including finance, identity management, and insurance contracts. Due to the need for repeatability, smart contracts can only access the current state of the blockchain. The use of external data may in fact result in a lack of consensus on the evolution of the system, thus prompting splits in network agreement. This would jeopardize repeatable execution of a smart contract across the entire network, undermining system stability. Consequently, the imperative for diverse solutions arises to establish trust in third-party information.

This paper introduces an advancement in oracle systems: a cognitive oracle that combines smart contract logic with verifiable computation capabilities. This combination creates a system for evaluating external data, enabling various dApps to leverage the benefits of Machine Learning as a Service (MLaaS). We deem this to be timely, as the EU AI Act and the US Executive Order on AI mandate that AI companies disclose technical details about their models [3,4]. Nevertheless, compliance is not easily achievable due to the rapidly evolving AI market, associated standards, resource constraints, and intellectual property issues. Part of this rapid regulatory evolution is the key advantage of MLaaS, which allows access to sophisticated ML models without the need for extensive infrastructure or expertise.

A significant challenge in deploying ML models is the black box problem, which refers to the lack of transparency in how they make decisions once they are in use. Recent studies highlight various approaches and advancements in making ML more explainable [5,6]. The financial implications of explainability in ML models are also non trivial. For instance, a study by [7] analyzed the cost of explainability in credit scoring models, revealing a difference of 15 to 20 basis points in annual return on investment between the best-performing black-box models and their explainable counterparts. This indicates a trade-off where opting for more transparent models may result in a slight decrease in ROI. In the context of an oracle system, this is crucial

☆ This article is part of a Special issue entitled: 'Blockchain for pervasive systems' published in Computer Communications.

* Corresponding author.
E-mail addresses: marco.esposito@polimi.it (M. Esposito), francesco.bruschi@polimi.it (F. Bruschi), donatella.sciuto@polimi.it (D. Sciuto).

https://doi.org/10.1016/j.comcom.2026.108467
Received 14 July 2024; Received in revised form 2 September 2025; Accepted 22 February 2026
Available online 5 March 2026

in guaranteeing that the ML model provider remains incentivized to adopt explainability measures.

Both private and public organizations consider explainability as a key requirement for AI systems, positively impacting other essential qualities such as their reliability and accountability [8]. Given that these qualities are also fundamental to existing oracle systems for smart contract data verification [9], incorporating explainable methods for ML model outputs in oracle inputs becomes a natural extension. However, explanation methods are susceptible to adversarial attacks. Research indicates that explainability itself can be manipulated, posing a risk to the integrity of MLaaS [10,11]. Verifiable computation solutions designed for explainability can help counteract these issues. In fact, ensuring correct explanations better aligns with the wider concept of ML verifiability, offering a more practical approach compared to pursuing complete explainability of inherently complex models [12]. In this regard, this paper draws on recent advancements in succinct ML verification protocols to prevent adversarial manipulation of the model during both the inference and post-hoc explanation phases within an oracle system.

Our contributions

We employ a well-documented methodology for outsourcing the cost of running ML inference backed by ML explanations in the form of class activation mapping, while being able to verify such computations at a variable cost. We discuss the feasibility of such an oracle architecture in orchestrating different economic incentives across various actors while minimizing opacity and undesired trust assumptions. Sections are organized as follows: Section 2 describes the state of the art, Section 3 presents our solution, Section 4 presents the experimental results along with prospective economic analyses, and Section 5 concludes.

2. State of the art

2.1. Background

Decentralized applications

dApps form a vast ecosystem, especially on EVM-compatible blockchains. Their logic is built using high-level languages like Solidity or Vyper, compiled into an ABI for interaction and then into EVM bytecode for execution by blockchain nodes. The applications are typically served over traditional web2-frontend infrastructure through HTTP requests, though efforts towards web3-native frontends are emerging. The organization of this code across multiple smart contracts represents a complex network underlying these applications, often analogous to traditional software systems both from an architectural and vulnerability perspective [13]. While the decentralized nature of these systems primarily relies on the consensus layer of the underlying blockchain, it also depends on other critical infrastructure components like decentralized storage systems (e.g., IPFS) and RPC node providers that are progressively being decentralized [14]. The dApp ecosystem has seen remarkable growth, expanding from around 2000 dApps in 2019 to over 5000 by August 2022, with daily active users exceeding 1.67 million [15]. This growth spans diverse sectors including decentralized finance, gaming, non-fungible tokens, data storage and provenance, privacy protection, and sharing economy applications.

Oracles

Oracles act as intermediaries between on-chain and off-chain environments, overcoming the limitation of dApps relying solely on blockchain data. These mechanisms enable smart contracts to interact with real-world information, bridging the gap between isolated blockchain systems and external events [16]. Data flow direction and initiation distinguish oracle types: inbound oracles bring external data to the blockchain, while outbound oracles send it out. Oracles operate in push mode (initiator sends data) or pull mode (initiator requests data) [17]. Oracle architectures are either centralized, relying on one trusted entity with single points of failure, or decentralized, using multiple nodes for added resilience but greater complexity [18]. Oracles offer solutions but pose challenges, especially in ensuring data trustworthiness in decentralized systems. Various data validation approaches have been introduced over the years:

- *Aggregation-based*: Implements predefined logic to derive deterministic results from multiple data sources
- *Staking-based*: Requires participants to stake assets as collateral against malicious behavior
- *Game-theory-based*: Creates economic incentives for honest behavior
- *Reputation-based*: Evaluates and tracks the reliability of data providers over time

Widely known implementations like Chainlink demonstrate the practical application of these concepts, employing a market-like approach through a decentralized network of nodes. These nodes compete to provide reliable data feeds, incentivized by reputation systems and economic rewards [19]. However, challenges still remain in balancing efficiency and resistance to various attack vectors like data manipulation [20]. More recent holistic approaches to addressing the oracle problem have been proposed, such as the Decentralized Truth Discovery Oracle model [21], which combines truth discovery techniques with Byzantine fault-tolerant consensus to extract data from multiple, potentially unreliable sources.

ML and blockchain

Machine learning extracts meaning from data using past performance to inform current and future decisions [22]. It encompasses supervised, unsupervised, semi-supervised, and reinforcement learning methods [23]. The synergy between ML and blockchain has long been recognized as a disruptive integration, ranging from secure data sharing marketplaces to data flow for explainable AI [24]. New conceptual architectures have emerged with promising uses in areas such as public health emergency management [25]. Many applications have emerged in federated learning and edge computing, where blockchain's decentralized nature complements ML's data processing capabilities while potentially mitigating centralization tendencies [26]. In Section 2.2, we explore ML's role in oracles, particularly in enhancing data veracity and semantic understanding.

Verifiable computation

Verifiable computation enables a computationally limited client (verifier) to delegate complex computations to a more powerful but potentially untrusted server (prover), while maintaining the ability to efficiently verify the correctness of returned results. This paradigm has seen renewed interest with the rise of blockchain scaling solutions, particularly in zk-rollups layer 2, where succinct proofs are used to verify the validity of batched transactions. Beyond blockchain scaling, verifiable computation has enabled numerous privacy-preserving applications like machine learning inference verification, proof-of-identity systems, and secure electronic voting while protecting sensitive information [27]. Our implementation leverages SNARKs (Succinct Non-interactive ARguments of Knowledge) as the primary cryptographic mechanism. In SNARK-based protocols, a prover generates a proof π_x demonstrating that applying function f to input x yields output y (i.e., $f(x) = y$). The defining characteristic of these proofs is their succinctness — verification complexity remains sublinear (sometimes constant) relative to the original computation time of $f(x)$. Implementation requires translating computations into arithmetic circuits composed of polynomial equations over large prime fields [28]. The number of constraints directly impacts the proof generation time and required computational resources. We build our oracle system upon Halo2, a specialized variant within the Polynomial Interactive Oracle Proof-based SNARKs family, which has been employed in various validity ML frameworks [29].

2.2. Related work

The majority of existing efforts in the oracle domain have concentrated on mechanisms for predicting or reporting data rather than checking its accuracy [9]. The challenge of verifying complex predicates that are difficult to express algorithmically has led to the development of specialized oracle systems with built-in dispute resolution capabilities.

Kleros [30] pioneered a decentralized justice approach where economic incentives and game theory guide juror behavior in resolving disputes through crowdsourced expertise. Building on this foundation, systems like ASTRAEA [31] and DeepThought [32] introduced formal frameworks for determining statement truth values through carefully structured voting mechanisms that combine staked capital with reputation scores. The need to address the gap between raw data verification and predicate evaluation is reflected on the thriving market of oracle platforms. UMA's optimistic oracle [33] uses a challenge-response mechanism for disputing semantic assertions. TrueBit [34] refined this with a verification game for subjective computations, while Reality.eth [35] introduced a flexible template-based system to standardize semantic queries. Integrating ML with oracle systems enables automated decision support with a human fallback. Recent studies have explored using ML models for initial assessments that can be challenged through jury-based appeals. For instance, MLBO [36] applies this to voter authentication with facial recognition as the first step, while maintaining human oversight for disputes. Similarly, the CONDOR protocol [37] connects AI oracles to smart contracts, adding an appeals process for contested algorithmic decisions.

Recent advances in verifiable ML have produced several approaches for ensuring computational integrity. We recognize their potential as building blocks for oracle systems tasked with automated semantic understanding that require verifiable outcomes.

Early work like SafetyNets [38] introduced interactive proof protocols for verifiable neural network execution. Subsequent approaches were typically constrained to specific architectures and scopes. For decision trees, zkDT [39] and Singh et al. [40] developed efficient verification protocols for model predictions. For convolutional networks, vCNN [41] and zkCNN [42] introduced specialized verification methods, with zkCNN optimizing proof generation through FFT-based protocols. Similarly, Mystique [43] provided efficient conversions between arithmetic/boolean circuits for ML computations, while ZEN [44] enabled verification of neural network accuracy through R1CS-friendly quantization. In contrast to architecture-specific approaches, newer frameworks like the one we adopted (detailed in Section 4.2) can handle arbitrary ML model types by converting any model's computational graph into a set of generic proving constraints [45]. The framework optimizes complex tensor operations into Einstein summations while constructing Halo2-compatible proving tables with parallel layouts, fixed-column lookups, and fixed-point arithmetic optimizations [46]. The setup produces a proving key pk and verification key vk through: $\text{Setup}(1^{\lambda}, W, f) \rightarrow (pk, vk)$, where W represents model weights and f denotes the architecture.

Related work specifically on querying oracles that leverage SNARKs explores batching techniques for general-purpose computation and storage to improve performance compared to monolithic circuit execution [47]. Another direction is to integrate traditional cloud interfaces and data processing technologies that combine real-time analytics, using protocols such as proof of SQL, where smart contracts verify query execution and data integrity in off-chain databases [48].

In the context of explainable ML, a key aspect is the degree to which humans can effectively understand and interact with MLaaS through causal explanations [49]. This human-centric property complements technical explainability to facilitate human–machine interaction, as proposed in our hybrid oracle systems. Explainable ML spans multiple dimensions including applicability, data types, and inner mechanisms [6]. Methods include post-hoc explanations for black-box models and inherently interpretable models, often trading accuracy for transparency, though recent work seeks verifiable interpretability without performance loss [50]. Early post-hoc techniques like LIME [51] and SHAP [52] approximate complex models with simpler, interpretable surrogates. In computer vision, methods focused on visualizing CNN features through deconvolutional networks [53] and gradient-based approaches [54]. Similar approaches relying on image feature removal strategies can produce out-of-distribution inputs, complicating the verification of whether the classifier relies solely on relevant features. For our image similarity scoring task, we employ a post-hoc visualization approach using activation maps and confidence scores rather than arbitrary feature removal, ensuring verifiable explanations by preserving learned representations and data distributions (see Section 3.4).

3. Proposed solution

3.1. Example problem: A hybrid model for carbon credits allocation

In this section, we introduce a novel oracle architecture that leverages verifiable ML to automate conditions typically requiring human intervention, like image analysis. This hybrid oracle architecture can be theoretically extended to a plethora of use cases. Healthcare institutions could implement verifiable diagnosis support, where ML models analyze medical imaging to detect anomalies, with human reviews available backed by ML explanations in case of disagreement. Similarly, classifiers could analyze sensor data and camera feeds to investigate autonomous vehicle incidents, with transportation safety experts reviewing the analysis if needed. In all such examples, the convenience of verifiable ML would be paired with the credibility of human reviewers, while the oracle's smart contracts provide the binding mechanism that makes these outcomes enforceable in a trustless environment.

In the environmental offsetting sector, carbon credits are a key method for counteracting emissions. Organizations use them to fund projects aimed at lowering carbon dioxide levels, adopting renewable energy, and supporting reforestation, which absorbs CO2 and produces oxygen. An entity can sponsor the planting of a particular area and receive credits, which may be recorded on a blockchain. To ensure the ongoing impact of these projects, it is crucial to monitor the growth and survival of the plants, verifying that the land remains dedicated to reforestation. This monitoring can be done using satellite imagery provided by various sources. Suppliers can then use signed, localized, and timed satellite images to verify vegetation growth and prevent damage.

The proposed solution leverages an off-chain classifier provided as a service by an environmental credit organization (Alice in Fig. 1). This classifier assesses user-submitted (e.g. quarterly) satellite images to determine eligibility for issuing new carbon credits by verifying the vegetation status on designated land parcels. Specifically, for each registered parcel, the classifier processes high-resolution (10m/pixel) satellite imagery to detect changes in vegetation density, unauthorized land use modifications, and potential health issues like disease patterns. The system compares these signed satellite images to previously approved photos (e.g. from past quarters) to confirm that vegetation levels are maintained or increased, with any detected large decrease (e.g. above 5%) automatically failing verification. This raises the question: where and by whom is the classifier executed? Running it on private infrastructure reintroduces challenges such as centralization, single points of failure, and potential manipulation — precisely the issues that blockchain technology aims to address. However, running the classifier on-chain is unfeasible due to computational limits and high costs. To resolve this, we integrate SNARKs into the oracle, allowing model providers to prove correct inference computations. Users can then claim carbon credit tokens proportional to their verified hectares once the classifier validates their submission with proof. In cases where users disagree with the ML output — for instance when seasonal variations might be misclassified as vegetation loss — they can request a jury-based oracle component for an impartial human review.

Rationale for verifiable ML explanations in dispute resolution

(1) *Information asymmetry*: The jury lacks direct insight into the model's decision-making process, which could lead to suboptimal verdicts. Post-hoc explanations can mitigate this asymmetry.

(2) *Strategic Manipulation*: The model providers may have incentives to manipulate explanations in two ways:

- False Reasoning: They could generate explanations that highlight irrelevant or misleading features to justify an incorrect classification.
- Adversarial Explanation: They could intentionally provide explanations showing apparent model bias or unreasonable feature attribution to discredit a correct classification.

In our cost analysis Section 4.4, we assume that providing ML explanations is rational to address information asymmetry. However, our goal is to explore whether it is economically viable for the model provider to generate a proof with cheap verification, or for each juror to compute the explanation individually. These are game-theoretical considerations that should take into account our social oracle component, as illustrated in Section 3.3. Overall, in the context of our example use case, this architecture aims to reduce reliance on external parties, thereby potentially cutting down both the costs and time involved in the acquisition or renewal of carbon credits.

3.2. Oracle lifecycle

In a complete scenario, for a landowner user to be rewarded with carbon credits after a period of vegetation growth, they need to interact with a web application (see dApps in 2.1) and undergo a verification process to confirm their eligibility. The oracle will leverage one of the pre-approved classifiers known to it (e.g., the smart contract component may hold cryptographic commitments to the classifiers' code) and follow the outlined steps in its lifecycle, including in the event of a dispute. The reader can refer to Fig. 1, where Bob represents the landowner user and Alice corresponds to the ML model compute provider. Their interactions with the oracle are illustrated at a high level, with the red numbers in the figure corresponding to the section numbers below. It should be noted that our prototype implementation in [55] focused primarily on the ML runtime prover and on-chain verification logic, while keeping the on-chain business smart contract economics and voting logic at a conceptual level.

3.2.1. Input submission

The user captures a new geotagged, timestamped satellite photo signed by a trusted source and uploads it to IPFS, obtaining the image's content-addressed hash h_{img}. The user then submits this hash and the trusted source's signature via the calldata of a designated smart contract function `validateImage()`. This function validates the authenticity of the trusted source's signature, assigns the hash to a mapping variable with previously submitted image hashes, and emits an event containing a unique request identifier and h_{img} for the off-chain classifier and prover execution.

3.2.2. ML runtime outcome

The user runs the binary classification task through API calls to Alice's MLaaS on the certified images corresponding to h_{img}, outputting a similarity score compared to the previous image. Regardless of whether the score is positive (indicating that the user has maintained vegetation growth) or negative, the system generates a SNARK of the correct classifier execution.

3.2.3. Smart contract verification

The smart contract is equipped to verify the proof with significantly less complexity than running the classifier itself. The deployed EVM verifier has the verifying keys baked into its code, enabling efficient proof verification via inline assembly. Besides the proof, the calldata of `verifySnark()` includes the instance, i.e. all the data shared between the prover and verifier. In the current implementation, the instance includes the public output similarity score but in a real-case scenario this would also include the image's content-addressed hash h_{img} and the unique request identifier to prevent replay attack or proof substitution.

3.2.4. Dispute appeal

Even if the proof is accepted by the on-chain verifier, the user and/or a supervisory entity may still disagree with the final similarity score. In such cases, a dispute appeal can be initiated by requesting a subsequent verdict from a jury (e.g. via function call `appeal()`), as described in Section 3.3. To reach an informed decision, the jurors may also issue an explanation request to the model provider.

3.2.5. ML runtime explanation

The model provider runs an explanation method over the disputed images using the committed model parameters. This produces a CAM visualization highlighting influential regions for the similarity score (Section 3.4). To ensure correct computation, the model provider generates a second SNARK via an aggregation scheme (see Fig. 4) that proves:

- The explanation matches the committed model and disputed images (h_{img})
- The explanation was computed according to the Score-CAM algorithm
- The same public parameters used in the original classification proof are maintained

Once this aggregate proof is submitted on-chain, the jurors can verify it in a trustless manner.

3.2.6. Verdict issuance

Based on the overall outcome of the votes cast via `castVote()` from both voters and certifiers (Table 1), the verdict execution can be handled through a governor contract implementation (not shown in the picture). Only if the jury supports the user's appeal, i.e. the proposal reaches TRUE state (see Table 1) by meeting quorum and majority thresholds, then an execution function mints the approved carbon credits to the credit issuance contract and marks them as claimable (via `transfer()`) by the verified user address.

Now let us consider the two components (social oracle and classifier) in more detail.

3.3. Social oracle

The proposed social oracle is inspired by the idea of ASTRAEA [31] and DeepThought [32] and has some theoretically researched changes that better adapt the characteristics of oracles to our context. It consists of judges who are called upon when a user files an appeal within the dApp. To ensure honest voting, a commit-reveal scheme is used, where the vote is secret during the commit phase and cannot be changed. It is crucial to note that any judge who declines to disclose their vote in the second phase of this procedure forfeits the complete stake. Assuming a batch of images submitted to the dApp, judges are categorized as either voters with randomly selected images, or certifiers with chosen images (Fig. 2). Both carry varying responsibilities and rewards: the former face lower risks and earn less, while the latter experience the opposite. Voters are not at liberty to decide on the images up for voting, but certifiers are. The stake must be set between

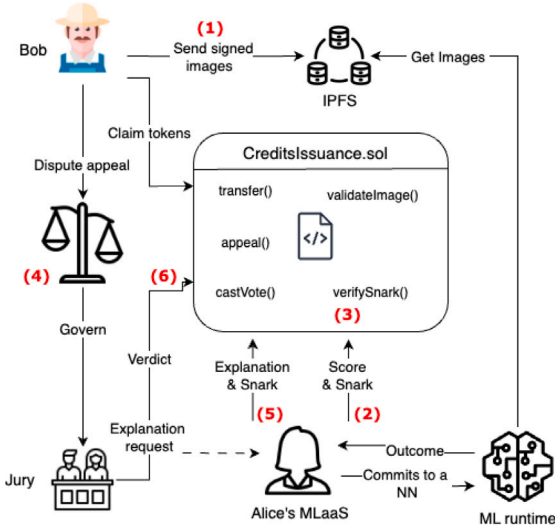

Fig. 1. High-level workflow of the oracle, combining ML inference and social dispute resolution.

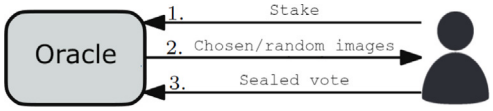

Fig. 2. Voter and certifier.

$$OUT_v = \begin{cases} TRUE, & SUM_{v,TRUE} > SUM_{v,FALSE} \\ FALSE, & SUM_{v,TRUE} < SUM_{v,FALSE} \\ Unknown, & SUM_{v,TRUE} = SUM_{v,FALSE} \end{cases}$$

Fig. 3. Voter Outcome.

Table 1
Oracle outcome.

OUT_c \ OUT_v	*TRUE*	*FALSE*	*Unknown*
TRUE	*TRUE*	*Unknown*	*Unknown*
FALSE	*Unknown*	*FALSE*	*Unknown*
Unknown	*TRUE*	*FALSE*	*Unknown*

a minimum of *min_stake_voter* and a maximum of *max_stake_voter* for the voter, and between *min_stake_certifier* and *max_stake_certifier* for the certifier. It is worth noting that *min_stake_certifier* is strictly higher than *max_stake_voter*, preventing any judge from filling both roles.

With respect to the considered example, the query pertains to the suitability of the satellite image submitted by the user. Specifically, does it portray the land area relevant to the user's plantation and does it exhibit an equal or greater amount of vegetation than the user's previously approved satellite image? To calculate the voter outcome OUT_v (Fig. 3), we note that $SUM_{v,TRUE}$ is the total weight of favorable votes received, while $SUM_{v,FALSE}$ is the aggregate weight of unfavorable votes. Additionally, the jury can also vote on the similarity between the two images based on the explanation provided by the explainability method detailed in Section 3.4.

The outcome of the OUT_c certifiers is calculated in the same way. Therefore, the final grade can be obtained by referring to Table 1.

In the event that the certifiers produce an unknown outcome, the oracle outcome is decided by the voters alone. As in DeepThought, the weight assigned to a judge's vote is determined by both their stake and reputation. Reputation is an integer ranging from 1 to *max_rep*

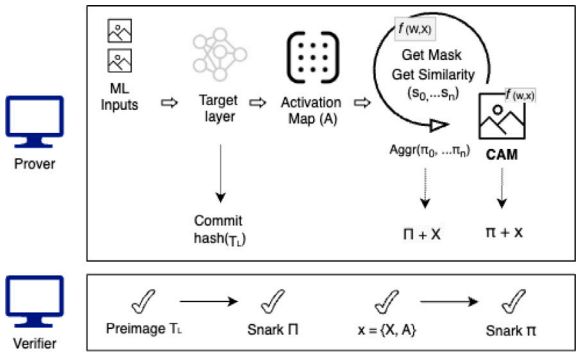

Fig. 4. Verifiable ML explanations.

and serves to indicate which judge is more trustworthy and therefore deserves more weight in the vote. The weight of the vote is depicted using the following formula:

$$f(s,r) = \sqrt{s}[\alpha\sqrt{r} + (1-\alpha)r]$$

where s represents the stake, r indicates reputation and α denotes a value between 0 and 1 that is determined during the jury setup phase to decide the weightage of each factor in the overall vote. Unlike in previous implementations like Deepthougth, for any α value, the stake's actual voting contribution is limited by a square root, preventing individuals with substantial economic resources from dominating the voting procedure, hence promoting greater decentralization. This function applies to both voters and certifiers. As far as rewards and penalties are concerned, there are two types: the first pertains to reputation, while the second relates to stakes, specifically economic rewards. In relation to reputation, judges earn a point if they vote correctly, i.e., if their vote matches the final verdict, but loose a point if they vote incorrectly. For what concern economic return, rewards and penalties are influenced by both the judge's stake and reputation. Whenever there is a vote, a bounty B is established, which the loser between the two parties is obliged to pay. Our proposal is that this bounty is initially split into two parts, B_v for the voters and B_c for the certifiers, in the following way:

$$B_v = 0.4 \cdot B \qquad B_c = 0.6 \cdot B$$

This is done to ensure that certifiers receive a greater portion of the reward, given their increased level of risk. Our proposed rewards and penalties are expressed in the following manner:

$$r_v = s + \frac{1}{2} \cdot B_v \left(\frac{s}{s_{\text{tot}}} + \frac{r}{r_{\text{tot}}} \right)$$

$$r_c = s + \frac{1}{2} \cdot B_c \left(\frac{s}{s_{\text{tot}}} + \frac{r}{r_{\text{tot}}} \right)$$

$$p_v = s - \frac{1}{2} \cdot B_v \left(\frac{s}{s_{\text{tot}}} + \frac{r}{r_{\text{tot}}} \right)$$

$$p_c = s - \frac{1}{2} \cdot B_c \left(\frac{s}{s_{\text{tot}}} + \frac{r}{r_{\text{tot}}} \right)$$

where the voter's reward, the certifier's reward, the voter's penalty, and the certifier's penalty are respectively denoted by r_v, r_c, p_v, and p_c. A reward differs from a penalty by adding or subtracting the two values that rely on the stake and reputation weighted against their totals. As noted earlier, the voter incurs less penalty and reward than the certifier. Note that the weightings for stake and reputation are equal in contributing to half the final value, with equal stake and reputation ratios.

3.4. Classifier component

We employed a custom SNN (siamese neural network), which is typically used to compare two images for tasks like signature identification and facial recognition. Unlike typical classification, it measures similarity between images rather than assigning labels. Practically, there are two identical subnetworks, $f(\theta, x)$, that share the same parameters θ. These subnetworks process two input images, x_1 and x_2, producing feature vectors $h_1 = f(\theta, x_1)$ and $h_2 = f(\theta, x_2)$. The similarity between the images is then measured by a distance function, typically the absolute distance:

$$d(h_1, h_2) = \sum_{i=1}^{n} |h_{1i} - h_{2i}|$$

where h_{1i} and h_{2i} are the components of the feature vectors h_1 and h_2, respectively.

The output of the network is a similarity score $\hat{Y}$ derived from the distance $d(h_1, h_2)$. This score is passed through fully connected layers to produce a final output, which is then fed into a sigmoid activation function to yield a probability $\sigma(\hat{Y})$ indicating whether the images are similar. In the one-shot learning setting, the aim is to minimize this distance for similar images and maximize it for dissimilar ones [56]. In our case this can be formulated as a binary classification problem using a Binary Cross-Entropy with Logits Loss:

$$\mathcal{L} = -\left[Y \cdot \log(\sigma(\hat{Y})) + (1 - Y) \cdot \log(1 - \sigma(\hat{Y}))\right]$$

where: Y is the binary ground truth label (1 for similar images, 0 for dissimilar images); $\hat{Y}$ is the predicted similarity score before applying the sigmoid function; $\sigma(\hat{Y}) = \frac{1}{1+e^{-\hat{Y}}}$ is the sigmoid function applied to $\hat{Y}$.

Following the training phase, our consideration extends to generating a SNARK for evaluating model execution and facilitating on-chain verification. SNARKs are integral to the emerging field of validity ML [57]. This process necessitates the involvement of two key parties: the *prover*, responsible for proof generation, and the *verifier*, tasked with validation. This ensures verifiers can validate proofs without revealing details or needing additional interaction with the prover post-generation and delivery [58]. In this oracle system, the prover corresponds to the entity utilizing the ML neural network, represented by Alice in Fig. 1. On the other hand, the verifier is embodied by a smart contract equipped with a set of methods triggered by user requests, specifically for obtaining carbon credits issuance. Additionally, a third party, typically the development team behind the dApp, plays a role in the overall setup of the architecture. This includes tasks such as circuit calibration and the generation of prover and verifier keys.

Class activation mapping

Our adopted method for providing an explanation of the ML inference process over images belongs to the broader class of CAM (Class Activation Mapping). These methods consist of visual explanations and traditionally rely on gradients to weigh the importance of activation maps. However, computing gradients in SNARKs, similarly to what is done in gradient descent during model training, can be rather costly due to the complexities of representing and verifying backpropagation operations. Instead, we adapted a method proposed by Wang et al. [59], where the importance of activation maps is determined without relying on gradients. This method, known as Score-CAM, computes the weight of each activation map by evaluating the increase in the score of the target class when the map is used as a mask on the input image. To work with our siamese neural network, we modified Score-CAM by using the similarity score directly as opposed to the target class score. Moreover, instead of targeting the last convolutional layer, we selected an earlier layer for lower-level feature extractions (i.e., basic shapes rather than high-level semantic features). By applying ReLU, negative values are discarded, making the saliency map more interpretable, namely the highlighted regions correspond to areas that positively impact the similarity score:

$$L_{\text{Score-CAM}} = \text{ReLU}\left(\sum_k \alpha_k A_l^k\right)$$

where A_l^k is the activation map of the kth channel at layer l, and α_k is the weight computed based on the increase in the similarity score using the softmax function:

$$\alpha_k = \frac{\exp(S_k)}{\sum_j \exp(S_j)}$$

Here, S_k is the similarity score obtained by passing the masked image through the network:

$$S_k = f(X_1 \odot \text{Up}(A_l^k), X_2) - f(X_b, X_2)$$

where $\text{Up}(A_l^k)$ upsamples the activation map to the input size, X_1 and X_2 are the input image pairs, X_b is a baseline image, and $\odot$ denotes element-wise multiplication.

Table 2
ML Inference in SNARKs (CPU total time and Keys sizes)

Model parameters	Optimized	Setup	Proving	Verification	Vkey	Pkey
23 365	Resources	775.63 s	1085.28 s	0.23	6 mb	20 gb
	Accuracy	1175.48 s	1895.1 s	0.48 s	12 mb	32 gb
463	Resources	44.14 s	56.87 s	0.04 s	339 kb	910 mb
	Accuracy	587.47 s	1096.52 s	0.24 s	5 mb	18 gb

4. Experimental evaluation

This section outlines our setup and the cryptographic protocols supporting verifiable computations. We evaluate the performance of the proposed hybrid oracle system by analyzing key performance indicators across its various architectural components. Traditional metrics, such as proof generation and on-chain verification costs, are reported in Section 4.3. Additionally, we present a broader evaluation framework in Section 4.4, incorporating economic efficiency and game-theoretic considerations.

4.1. Models configuration

We implemented a specific configuration setup to verify the costs of using a classifier in the described context [55]. Two custom SNN (siamese neural networks) were designed for image similarity tasks. Both networks follow a similar architecture, comprising convolutional layers followed by pooling layers to learn meaningful image embeddings. The feature extraction phase for both models consists of convolutional and max-pooling layers, and the fully connected section includes a single linear layer with dropout. The two models differ in the number of convolutional filters and the complexity of their architectures. The first model contains 23,365 parameters, whereas the second model contains 463 parameters. Given the simplicity of the image similarity task at hand, they proved to be adequate despite their limited complexity and parameter count. The training procedure was based on a dataset of 1000 samples, encompassing pairs of synthesized satellite images, alongside binary labels indicating the similarity relationship (i.e., 0 = vegetation was altered; 1 = vegetation grew or remained stable). Both SNNs were trained using binary cross-entropy with logits loss and stochastic gradient descent, exploring various learning rates. It is important to note that the primary objective was not to achieve high model accuracy and reliability, but rather to exemplify and evaluate a scenario involving two distinct model sizes. This also served to demonstrate that the overall verifier cost can maintain a relative independence from the model size.

Table 3
On-chain verification costs and accuracy.

Model parameters	Optimized	Deploy (Gas - Eth)	Call method (Gas - Eth)	SNN output[a]	Circuit output[b]	Mean percent error[c]
23 365	Resources	2 499 001 – 0.017	746 072 – 0.005	0.074	0.0335	0.11
	Accuracy	3 752 940 – 0.026	1 120 319 – 0.008		0.072	0.0025
463	Resources	2 341 754 – 0.016	682 362 – 0.005	0.604	0.599	0.046
	Accuracy	3 429 956 – 0.025	1 000 685 – 0.007		0.603	0.029

Note: Resources and Accuracy rows correspond to different circuit logrow parameters (16 and 21, respectively). Larger models incur higher setup and proving times, with a significant increase in proving key size. Local verification remains efficient (<0.5 s), and on-chain verification, calculated at 7 gwei per gas unit, remains relatively inexpensive regardless of model size and optimization. Better optimization leads to minimal circuit quantization error, but the storage and communication overhead for the prover is significant at 32 GB.

[a] $SNN\_output \approx sigmoid(logit)$.

[b] $Circuit\_output \approx sigmoid(quantized\_output/2^{scale\_multiplier})$.

[c] $mean((logit - (raw\_output/2^{scale\_multiplier}))/logit)$.

4.2. Verifiable computation protocol

Proving ML inferences

We employed the EZKL framework (version 11.4.1) [60], which includes tools to generate the zk-SNARK. This framework is specifically designed for conducting inference on deep learning models within the Halo2 proof system, implemented in Rust. EZKL allows users to express computations as a graph, which facilitates the creation of:

1. A prover capable of generating proofs validating that, when provided with a specific input, the computations yield a predetermined output; 2. A verifier capable of validating proofs, designed to operate within an Ethereum Virtual Machine (currently implemented as a Solidity contract).

The setup involves defining proof criteria, exporting the neural network to an ONNX file (with optional tuning), and using KZG commitments. This process generates circuit settings, proving, and verification information. Once configured, the neural network circuit requires only a single setup, making subsequent proof generation a routine task, including the multiple inferences required for constructing ML visual explanation (see Section 3.4). Therefore, proving and verifying key are not tied to individual data points and deploying the generated Verifier.sol contract is a one-time task for a specific circuit. For a given model, the proof time complexity is determined by the number of constraints, n_{con}, which generally scales linearly with the number of operations (e.g., MACs or FLOPs) in the model inference. Thus, the proof generation complexity can be approximated as $O(n_{con})$ [45]. This ensures scalability even with larger and different model architectures, as it has been successfully tested [61]. The EZKL framework provides both Rust CLI and Python bindings. Here, we utilized the latter in a Jupyter environment. All computations were performed on a MacBook M2 (8-core CPU) with 24 GB of RAM.

Proving ML explanations

Checking the correct computation of neural network visual explanations can be conceptualized as a two-step protocol. The cost of both steps falls on the prover of the original ML inference for carbon credit issuance, although one could envision an additional actor, an auditor, whose job can be subsidized by the oracle DAO.

4.2.1. Outside SNARKs

"Activation maps attestation" Hook into the committed target layer to capture activation maps and resize them to the original input size. This computation is relatively lightweight ($O(s)$, where s is the size of the activation maps) and can be easily verified by attesting its hash on-chain. A commitment to the requested target layer should correspond to the derived activation maps' hash. Subsequently, using the extracted activation maps, generate masked inputs by iterating over their channels.

Table 4
Explaining ML in SNARK (CPU total time)

Onnx graph	Setup[a]	Proving[b]	Aggr setup	Aggr proofs	Verification
Multiple inferences	n/a	168.57 s	1483.69 s	7950.2 s	0.44 s
CAM computation	21.28 s	41.89 s	n/a	n/a	0.09 s

[a] New setup keys are only required for CAM computation.

[b] Proving is cumulative for the Multiple inference SNARKs.

4.2.2. Inside SNARKs

"Multiple inferences" Perform multiple forward passes with the perturbed inputs (i.e., masks) through the model to obtain the corresponding outputs, technically to obtain the list of logits from the formula in Section 3.4:

$$\text{logits} = \{S_k \mid S_k = f(X_1 \odot \text{Up}(A_l^k), X_2) - f(X_b, X_2), \forall k\}$$

In Halo2, proof aggregation can reduce the size of the aggregated proof compared to summing all individual proofs. However, the complexity of aggregation time does not scale linearly with the number of proofs. Instead, it scales with a logarithmic factor relative to the size of the circuit. So, with the generation of repeated model inference proofs, the original complexity is $O(kmp)$, where k is the number of computed masked inputs, m is the number of network layers, and p is the size of the input per layer. This now turns into $O(kPolyLog(mp))$. By using a sufficiently large Common Reference String (CRS), we integrate everything into a meta-proof Π. As for the final "CAM generation", we operate a softmax function over the above list of logits to get the weights $\{\alpha_k\}$, and multiply the committed activation maps by these weights to generate the final CAM. This is a single proof generation dependent on the size of its inputs (i.e., activation maps and the number of logits).

The verifier, having access to the meta-proof Π and the proof π along with their corresponding instances, performs the following operations either locally or on-chain: (i) checks the validity of the meta-proof Π; (ii) ensures that the activation map used for π matches the previously committed activation map. This is done by checking that the hash of the chosen target layer that extracts the public activation map found in the proof file corresponds to the hash committed on-chain; (iii) the verifier also examines the single proof π of CAM generation to ensure that it incorporates the outputs of the meta-proof Π as well as the committed activation map as public instances. This involves ensuring that the final CAM is correctly derived from the weighted activation maps. The comprehensive verification process can expressed as:

$$V(\Pi) \wedge V(\pi) \wedge (H(T_l) = T_l^{\text{committed}}) = \text{True}$$

where $V(\Pi)$ and $V(\pi)$ are the verification functions and $H(T_l)$ is the hash of the target layer T_l. A schematic depiction of the verifiable ML visual explanations can be found in Fig. 4.

Table 5
Explaining ML in SNARK (Storage overhead)

Onnx graph	Vkey	Pkey	Proof(s)	Aggr Vkey	Aggr Pkey	Aggr Proof	–
Multiple inferences	n/a	n/a	243 kb	2.11 mb	35.44 gb	18 kb	–
CAM computation	223 kb	763.6 mb	2.1 mb	n/a	n/a	n/a	–

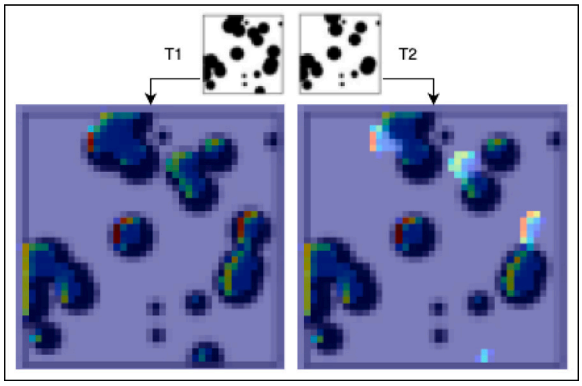

Fig. 5. Test satellite images with overlaying CAM.

4.3. Execution results

The following results are presented separately for the prover and the verifier, focusing on three metrics: the trade-off between circuit accuracy and computational overhead, storage efficiency as reflected in the sizes of proving and verification keys, and on-chain verification costs, which directly impact economic feasibility. Notably, while higher-precision circuits improve accuracy, they significantly increase computational and storage demands, making this trade-off crucial for practical deployments.

The optimization factor primarily depends on the logrows parameter, which indicates the number of rows pre-allocated for the Halo2 circuit matrix to represent the neural network. In Table 2 and 3, *resources* and *accuracy* respectively mean that logrows parameters of 16 and 21 were used to calibrate the circuit, resulting in 2^{logrows} matrix rows. The assessments are hence conducted under dual setups, emphasizing either CPU and memory optimization or accuracy refinement through increased precision in fixed-point representation. The setup phase can be conducted by the prover or the development team, and then share both the proving pk and verification key vk.

Prover

The input images tested here are an example of two dissimilar representations of the vegetation growth, specifically the second image of Fig. 5 depicts a reduced number of plants. As Table 3 shows, the output for the larger model correctly identified them as such (i.e. sigmoid logit < 0.5), whereas the smaller model score was just above the similarity threshold. The last three columns of Table 3 compare the class probability figures of the SNN, circuit representation, and the average error rate of circuit output against class predictions calibrated across 20 input images. This reflects the average quantization error stemming from the transition of floating-point to fixed-point representations within the circuit, as it operates over field elements. Table 2 catalogues the CPU total time required for keys and proof generation for a singular witness test. As for the storage requirements these are evidently higher for the prover especially when creating a proving key of a circuit with more constraints - i.e. optimized for accuracy.

Verifier

These are the results obtained by cryptographically verifying on-chain that a specific SNN with fixed model parameters, i.e., committed for all proofs, and with the output subsequently made public on-chain. Table 3 displays the costs acquired from testing the smart contracts deployment and executing a carbon credits claim call with on-chain verification of the test using a Hardhat local development environment. The economic evaluation is pegged to a conversion metric of 7 gwei per gas unit when translating gas into ETH. These costs might be reasonable with current ETH evaluations, but they are unsustainable in the long term. However, with EVM compatibility for the verifier contract, they could be reduced through the use of L2 solutions. Overall, the on-chain verification process may assure users of the integrity of the ML service's output with their image input. However, skepticism may arise in ambiguous situations, such as the current test inputs. For instance, the carbon credit issuance smart contract might stipulate that credits cannot be claimed if the quantization error is below a critical threshold. When the quantization error is significant, human oracle intervention along with ML explanations may be necessary for dispute resolution.

CAM generation

For practical purposes, tests were run only on the smaller SNN optimized for resources. The single convolutional layer present in the model was the target layer from which to extract a three-channel activation map. The CPU total time for a complete CAM generation was around 4 s and this cost is discussed in the next Section 4.4. Fig. 5 shows our two test inputs with the visual explanation on top, where the pixels tending to red show the parts of more meaningful feature extraction. Regarding the costs associated with running the Halo2 proof system, we observe that the largest computational overhead is contributed by the proofs aggregation step (Table 4), while the cost of local verification remains quite low. Table 5 also shows that the proving key size of over 35 GB is the main limiting factor for memory handling and storage. This size constraint seems to pose a challenge for scaling SNARKs to accommodate larger neural networks.

4.4. Oracle cost analysis

This section explores how metrics like dispute frequency thresholds and sunk costs from automated response verification indicate system viability. Moreover, the game-theoretic analysis of jury composition and stake-reputation dynamics offers insights into incentive alignment and risks. Future work should build on these indicators to establish standardized benchmarks for evaluating hybrid oracles.

To address the economic convenience of using the ML service instead of a fully jury-based oracle in terms of cost, we can analyze the following inequality to be true:

$$(k+1) \cdot n \cdot j < \sum_{t=0}^{n} Y_t$$

where: - j is the cost (i.e., bounty) for a juror, including both certifiers and voters as described in Section 3.3, - n is the number of jurors, - Y_t is the cost of running a proof of ML inference at time t, - k is the number of disputes resolved by the jury, - $\sum_{t=0}^{n} Y_t$ represents the total cost of generating ML inference proofs over a time period. Assuming n and j are fixed, using our hybrid oracle will be economically beneficial only if the frequency of dispute appeals remains under a certain threshold - i.e., jurors do not need to be rewarded as often, thereby reducing the overall costs. To put this into perspective, if disputes are infrequent, the cost incurred from rewarding the jurors $(n \cdot j)$ will be less than the cumulative cost of having a ML provider prove the inference $(\sum_{t=0}^{n} Y_t)$. Therefore, the primary economic benefit of using an ML-based oracle lies in its ability to reduce the frequency of costly jury-based dispute resolutions.

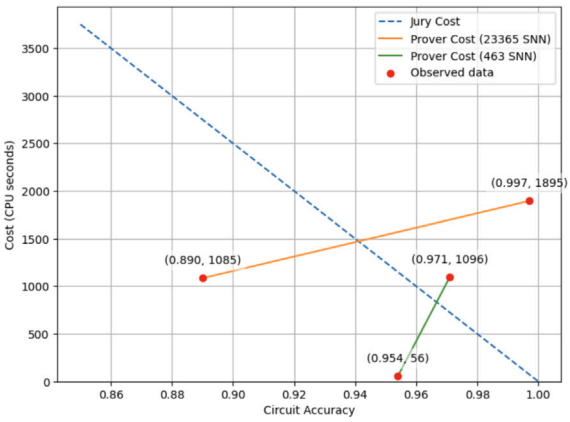

Fig. 6. Break-even cost simulation for proving ML inference versus human jury costs. The dashed blue line represents decreasing jury costs with improved circuit accuracy. Solid lines show prover costs for two different model architectures (23,365 and 463 parameters), with red dots indicating observed measurements (accuracy, cost). The intersections identify accuracy levels where ML-based verification becomes more cost-effective than jury verification. Accuracy is shown as the inverse of mean percent error. Cost is measured in CPU seconds.

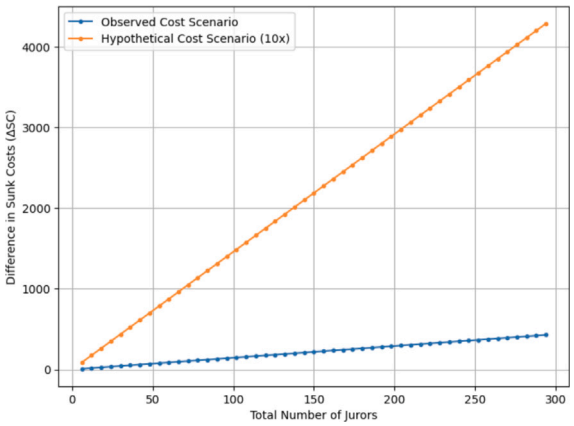

Fig. 7. Difference in expected sunk costs (ΔSC) between local execution and SNARK verification as jury size increases. The blue line shows the observed scenario with local execution cost $j_e = 4$ and proof verification cost $j_v = 0.5$ CPU seconds, while the orange line represents a hypothetical high-cost scenario (10x higher computational costs). Both scenarios assume a 1:5 voter-to-certifier ratio and 50% probability of certifiers incurring sunk costs.

While n and j could be thought of as fixed parameters in a DAO, Y_t is likely to vary based on ML provider's internal decisions to use circuits with different optimizations. Therefore, a probabilistic model could be used to estimate the likelihood of disputes as a function of the circuit accuracy, namely the fidelity of the Halo2 circuit output calculated as $(1 - |\text{mean percent error}|)$. Let $P(D|A)$ represent the probability of a dispute D given accuracy A. In practice, this estimate would require sufficient historical data on disputes. Here we simply assume $P(D|A)$ decrease monotonically with increasing A. Once above a certain threshold, the left side of the inequality would decrease, representing fewer jury involvements and hence lower total cost. However, this is incomplete as it would leave out the increased costs for the prover when A increases, as shown in Tables 2 and 3. To account for this, we must use $C(A_t)$ as the total cost of proving ML inference over a circuit with accuracy A_t at time t. The final inequality would then look like:

$$(N \cdot P(D|A) + 1) \cdot n \cdot j < \sum_{t=0}^{n} C(A_t)$$

where N replaces k as the total number of oracle queries over the analyzed period. To visualize at what point increasing the accuracy of the model would be most cost-effective, we can approximate $P(D|A)$ and $C(A)$ based on our limited data points. For simplicity, $C(A)$ is linearly interpolated between our known data points. The intersection of these curves represents the optimal accuracy level where the total cost for the oracle is minimized. Fig. 6 illustrates these break-even points with a fixed number of 50 jurors, each costing an equivalent of 100 CPU seconds, over 5 total individual oracle queries processed. Notably, having the break-even points within the range of our observed data means we do not rely on extrapolation, making the estimate more realistic. However, using a linear model for $P(D|A)$ simplifies the analysis and may not capture the complexity of real-world scenarios where such relationship could be non-linear.

Economics of ML explanations

This is the case of a dispute appeal when the jury requests explanation of the model decision to better inform its final verdict. Let Y' be the cost of running the proof generation of CAM. Because of the complexities of generating a SNARK of the model explanations (as described in the above sections), we can assume Y' to be always greater than Y alone. Then $\Delta Y = |Y' - Y|$ will be the added cost borne by the prover in the event of a dispute appeal. Let j_e be the cost for a juror to run the ML explanation themselves, and j_v be the cost for a juror to verify a handed proof. Based on our data, $j_e \approx 4$ and $j_v \approx 0.5$ CPU seconds. The economic benefit of having the ML provider generate a SNARK of the visual explanation, rather than having each juror run it on their devices, becomes more pronounced with an increasing number of jurors and disputes. This is because the cost of generating a single SNARK proof can be amortized over all the jurors and over multiple disputes. Intuitively this should results in:

$$\Delta Y < n \cdot (j_e - j_v)$$

Considering the cost of aggregating the proof for multiple inferences as shown in Table 4, and using our estimates of j_e and j_v, we can calculate the number of jurors required to make this inequality true:

$$n > \frac{\Delta Y}{j_e - j_v} = \frac{\Delta Y}{3.5}$$

For example, if $\Delta Y = 7883$ CPU seconds the number of jurors required would be approximately 2250. Such a large number of jurors is impractical for most oracle systems, highlighting the uneconomical nature of having to compensate the ML provider for explaining ML in SNARK.

Sunk costs and jury composition

It is safe to assume that whether running the ML visual explanation or verifying its proof does not affect the juror's decision-making process or the quality of visual information they receive. Regardless of the method, the ML explanation received could add little to zero information to their belief system. If so, jurors could have avoided the associated cost, denoted as j_e for running the explanation locally and j_v for proof verification only. Thus, having spent 4 CPU seconds for j_e and not 0.5 CPU seconds for j_v represents a *sunk cost* - i.e., a hidden economic disadvantage. The difference in potential sunk cost likely strengthens the case for explaining ML in SNARKs, as it reduces the magnitude of the sunk cost when jurors consider the provided explanation uninformative.

To have a more complete picture of this, we should distinguish between voters and certifiers, where the latter are assumed to be more conservative in their behavior (see Section 3.3). Let p_c be the probability that a certifier is convinced by the explanation and decides to increase their stake, and p_v be the probability that a voter makes a decision based on the explanation. We can reasonably assume that $p_c < p_v$ due to the certifiers' higher stakes and more cautious approach.

The expected sunk cost for certifiers (SC_c) can be expressed as: $SC_c = (1-p_c)\cdot(n_c \cdot j_e)$ when running the explanation, and $SC'_c = (1-p_c)\cdot(n_c \cdot j_v)$ when verifying the proof. Similarly for voters (SC_v): $SC_v = (1-p_v)\cdot(n_v \cdot j_e)$ when running the explanation, and $SC'_v = (1-p_v)\cdot(n_v \cdot j_v)$ when verifying the proof. Where n_c and n_v are the number of certifiers and voters respectively. The total expected sunk cost for the oracle system is: $SC_{total} = SC_c + SC_v$ when running the explanation and $SC'_{total} = SC'_c + SC'_v$ when verifying the proof. The difference in total sunk costs, $\Delta SC = SC_{total} - SC'_{total}$, represents the potential savings in sunk costs by having the jury verify visual explanation proof: $\Delta SC = (1-p_c)\cdot n_c \cdot (j_e - j_v) + (1-p_v)\cdot n_v \cdot (j_e - j_v)$. This formula highlights that the benefit of using SNARKs for ML explanation in terms of reducing sunk costs is more pronounced when: (i) the proportion of certifiers (n_c) is higher, as they are less likely to have strategic certainty (p_c is lower); (ii) the difference between local execution cost and SNARK verification cost ($j_e - j_v$) is larger. To better visualize these implications, Fig. 7 illustrates an observed cost scenario and a hypothetical cost scenario with 10 times higher computational intensity for both j_e and j_v. The plot shows that, assuming a 1:5 voter-to-certifier ratio, whereby certifiers only have a 0.5 chance of incurring sunk costs, the difference in sunk costs increases linearly with the number of jurors. This growth is particularly evident because SNARK systems can achieve sublinear proof verification time, demonstrating significant cost savings even with a higher computational load.

Overall, these estimates suggest complex and multifaceted economic dynamics for maintaining a sustainable hybrid oracle system. Several technical limitations could impact real-world scalability. As model complexity grows, the computational overhead of generating proofs for both inference and explanations could become prohibitive, especially if dispute frequencies remain high. These resource demands may restrict ML architectures, requiring trade-offs between model complexity and oracle efficiency. Future research should explore proof system optimizations and alternative architectures that can better balance the computational demands with the need for reliable data verification and semantic understanding.

5. Conclusions

We addressed the challenge of defining an oracle for applications requiring the extraction of complex and non-algorithmically definable features, such as the semantic content of an image. Our focus was on a realistic application scenario where carbon credit tokens are awarded for planting trees for CO_2 capture. We noted the limitations of current jury-based oracles, which are costly and suited for one-time evaluations. The proposed oracle design merges a classifier with a social layer, enabling users to request an informed review if they dispute the automated decision. Our experimental work investigated employing zk-SNARKs to provably preserve the classifier's integrity. An increasing number of studies have proposed new paradigms for verifying ML model inferences [41,43,62,63] and more recently even ML model training [64–66]. Utilizing the Halo2-based Ezkl toolkit, we confirmed the feasibility of validating proofs on the Ethereum blockchain at acceptable gas expenses, consistent with the results achieved in verifying even larger models in generative AI [61]. Additionally, to date, we are not aware of studies evaluating verifiable computations for ML explanations. Our findings suggest that current technology, particularly the aggregating circuit, still requires refinement to be competitive for scalable oracle solutions, due to its known higher memory footprint [45]. Future work should explore new approaches to reduce the overhead of such proofs of explanations and ensure that dispute resolutions can be informed efficiently by succinct proofs alone. As noted elsewhere, "Until users have real control over how algorithms behave, something is missing in current AI solutions. This causes massive distrust in AI, and apathy towards AI ethics solutions" [67]. On a positive note, considering ongoing advancements in proof system efficiencies reported in scholarly research, our oracle's framework might hold the promise of practical application in the short-medium term.

CRediT authorship contribution statement

Marco Esposito: Writing – review & editing, Writing – original draft, Visualization, Methodology, Formal analysis, Data curation, Conceptualization. **Francesco Bruschi:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Donatella Sciuto:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://github.com/okrame/xAI-oracle.

References

[1] Fran Casino, Thomas K. Dasaklis, Constantinos Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, Telemat. Inform. (ISSN: 0736-5853) 36 (2019) 55–81, http://dx.doi.org/10.1016/j.tele.2018.11.006, URL https://www.sciencedirect.com/science/article/pii/S0736585318306324, (visited on 11/30/2023).

[2] Weisheng Lu, et al., Exploring smart construction objects as blockchain oracles in construction supply chain management, Autom. Constr. (ISSN: 0926-5805) 129 (2021) 103816, http://dx.doi.org/10.1016/j.autcon.2021.103816, URL https://www.sciencedirect.com/science/article/pii/S0926580521002673, (visited on 11/30/2023).

[3] Tambiama Madiega, Artificial intelligence act, in: European Parliament: European Parliamentary Research Service, 2021.

[4] Joseph R. Biden, Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023.

[5] Dang Minh, H. Xiang Wang, Y. Fen Li, Tan N. Nguyen, Explainable artificial intelligence: a comprehensive review, Artif. Intell. Rev. (2022) 1–66.

[6] Truong Thanh Hung Nguyen, A Survey on explainable artificial intelligence: techniques, xai-based model improve-ment methods, applications.

[7] Jean Dessain, Nora Bentaleb, Fabien Vinas, Cost of explainability in AI: An example with credit scoring models, in: World Conference on Explainable Artificial Intelligence, Springer, 2023, pp. 498–516.

[8] Nagadivya Balasubramaniam, Marjo Kauppinen, Antti Rannisto, Kari Hiekkanen, Sari Kujala, Transparency and explainability of AI systems: From ethical guidelines to requirements, Inf. Softw. Technol. 159 (2023) 107197.

[9] Giuseppe Antonio Pierro, Honore Mahugnon, An analysis of the oracles used in ethereum's blockchain, in: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER, IEEE, 2023, pp. 878–885.

[10] Juyeon Heo, Sunghwan Joo, Taesup Moon, Fooling neural network interpretations via adversarial model manipulation, Adv. Neural Inf. Process. Syst. 32 (2019).

[11] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, Himabindu Lakkaraju, Fooling lime and shap: Adversarial attacks on post hoc explanation methods, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 180–186.

[12] K.R. Patil, Verifiability as a Complement to AI Explainability: A Conceptual Proposal [Preprint][Electronic resource].

[13] Sabrina Aufiero, Giacomo Ibba, Silvia Bartolucci, Giuseppe Destefanis, Rumyana Neykova, Marco Ortu, The network structure of smart contracts in ethereum dapps, Complex Networks 2023 (2023).

[14] Infura and DIN Team, The decentralized infrastructure network (DIN): A permissionless marketplace for high-throughput blockchain APIs and services across N-networks, 2024, Operational Draft, November 2024. (Accessed 14 November 2024), https://drive.google.com/file/d/1hCHmcXMN6YpmGQkdxSTuZb6Ne_EaehJt/view.

[15] Peilin Zheng, Zigui Jiang, Jiajing Wu, Zibin Zheng, Blockchain-based decentralized application: A survey, IEEE Open J. Comput. Soc. 4 (2023) 121–133.

[16] Yinjie Zhao, Xin Kang, Tieyan Li, Cheng-Kang Chu, Haiguang Wang, Toward trustworthy DeFi oracles: Past, present, and future, IEEE Access (ISSN: 2169-3536) 10 (2022) 60914–60928, http://dx.doi.org/10.1109/ACCESS.2022.3179374, URL https://ieeexplore.ieee.org/document/9785807/, (visited on 09/28/2023)..

[17] Kamran Mammadzada, Mubashar Iqbal, Fredrik Milani, Luciano García-Bañuelos, Raimundas Matulevičius, Blockchain oracles: A framework for blockchain-based applications, in: Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18, Springer, 2020, pp. 19–34.

[18] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, Davor Svetinovic, Trustworthy blockchain oracles: Review, comparison, and open research challenges, IEEE Access (ISSN: 2169-3536) 8 (2020) 85675–85685, http://dx.doi.org/10.1109/ACCESS.2020.2992698, URL https://ieeexplore.ieee.org/abstract/document/9086815, (visited on 11/18/2023), Conference Name: IEEE Access.

[19] Lorenz Breidenbach, Christian Cachin, Alex Coventry, Steve Ellis, Ari Juels, Andrew Miller, Brendan Magauran, Sergey Nazarov, Alexandru Topliceanu, Fan Zhang, Benedict Chan, Farinaz Koushanfar, Daniel Moroz, Florian Tramer, Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks.

[20] Mubashar Iqbal, Alessandro Chiarelli, Raimundas Matulevičius, Bridging two worlds: Framework for secure implementation of blockchain oracles, in: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering-Companion (SANER-C), IEEE, 2024, pp. 12–22.

[21] Yang Xiao, Ning Zhang, Wenjing Lou, Y. Thomas Hou, A decentralized truth discovery approach to the blockchain oracle problem, in: IEEE INFOCOM 2023-IEEE Conference on Computer Communications, IEEE, 2023, pp. 1–10.

[22] Devanshi Dhall, Ravinder Kaur, Mamta Juneja, Machine learning: A review of the algorithms and its applications, in: Pradeep Kumar Singh, Arpan Kumar Kar, Yashwant Singh, Maheshkumar H. Kolekar, Sudeep Tanwar (Eds.), Proceedings of ICRIC 2019, in: Lecture Notes in Electrical Engineering, Springer International Publishing, ISBN: 978-3-030-29407-6, 2020, pp. 47–63, http://dx.doi.org/10.1007/978-3-030-29407-6_5.

[23] Konstantin D. Pandl, Scott Thiebes, Manuel Schmidt-Kraepelin, Ali Sunyaev, On the convergence of artificial intelligence and distributed ledger technology: A scoping review and future research agenda, IEEE Access (ISSN: 2169-3536) 8 (2020) 57075–57095, http://dx.doi.org/10.1109/ACCESS.2020.2981447, Conference Name: IEEE Access, URL https://ieeexplore.ieee.org/abstract/document/9039606, (visited on 11/30/2023).

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

[25] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Blockchain and AI-based solutions to combat coronavirus (COVID-19)-like epidemics: A survey, IEEE Access 9 (2021) 95730–95753.

[26] Youyang Qu, Shiva Raj Pokhrel, Sahil Garg, Longxiang Gao, Yong Xiang, A blockchained federated learning framework for cognitive computing in industry 4.0 networks, IEEE Trans. Ind. Inform. 17 (4) (2020) 2964–2973.

[27] Ryan Lavin, Xuekai Liu, Hardhik Mohanty, Logan Norman, Giovanni Zaarour, Bhaskar Krishnamachari, A survey on the applications of zero-knowledge proofs, 2024, arXiv preprint arXiv:2408.00243.

[28] Anca Nitulescu, Zk-SNARKs: A gentle introduction, Éc. Norm. Super. (2020).

[29] Zhibo Xing, Zijian Zhang, Jiamou Liu, Ziang Zhang, Meng Li, Liehuang Zhu, Giovanni Russello, Zero-knowledge proof meets machine learning in verifiability: A survey, 2023, arXiv preprint arXiv:2310.14848.

[30] Clement Lesaege, Federico Ast, William George, Kleros, 2019, (Accessed 27 October 2023), https://kleros.io/static/whitepaper_en-8bd3a0480b45c39899787e17049ded26.pdf.

[31] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, Anastasia Kastania, Astraea: A decentralized blockchain oracle, in: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, ISBN: 978-1-5386-7975-3, 2018, pp. 1145–1152, http://dx.doi.org/10.1109/Cybermatics_2018.2018.00207, URL https://ieeexplore.ieee.org/document/8726819/.

[32] Marco Di Gennaro, Lorenzo Italiano, Giovanni Meroni, Giovanni Quattrocchi, DeepThought: a reputation and voting-based blockchain oracle, 2022, http://arxiv.org/abs/2209.11032, arXiv:2209.11032, [cs], (visited on 09/24/2023).

[33] U.M.A. Protocol, UMA documentation, 2024, (Accessed 14 November 2024), https://docs.uma.xyz/.

[34] Jason Teutsch, Federico Kattan, Blane Sims, Unchained report 2024, 2024, (Accessed 14 November 2024), https://truebit.io/unchained-report-2024.pdf.

[35] Reality.eth, Reality.eth whitepaper, 2024, (Accessed 14 November 2024), https://reality.eth.link/app/docs/html/whitepaper.html.

[36] Asmae El Fezzazi, Amina Adadi, Mohammed Berrada, Towards a blockchain based intelligent and secure voting, in: 2021 Fifth International Conference on Intelligent Computing in Data Sciences (ICDS), 2021, pp. 1–8, http://dx.doi.org/10.1109/ICDS53782.2021.9626751, URL https://ieeexplore.ieee.org/abstract/document/9626751, (visited on 11/29/2023).

[37] Bernardo Sata, Aizar Berlanga, Caroline P.C. Chanel, Jérôme Lacan, Connecting AI-based oracles to blockchains via an auditable auction protocol, in: 2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), 2021, pp. 23–24, http://dx.doi.org/10.1109/BRAINS52497.2021.9569808, URL https://ieeexplore.ieee.org/abstract/document/9569808, (visited on 11/30/2023).

[38] Zahra Ghodsi, Tianyu Gu, Siddharth Garg, Safetynets: Verifiable execution of deep neural networks on an untrusted cloud, Adv. Neural Inf. Process. Syst. 30 (2017).

[39] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, Dawn Song, Zero knowledge proofs for decision tree predictions and accuracy, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 2039–2053.

[40] Nitin Singh, Pankaj Dayama, Vinayaka Pandit, Zero knowledge proofs towards verifiable decentralized ai pipelines, in: International Conference on Financial Cryptography and Data Security, Springer, 2022, pp. 248–275.

[41] Seunghwa Lee, Hankyung Ko, Jihye Kim, Hyunok Oh, Vcnn: Verifiable convolutional neural network based on zk-snarks, IEEE Trans. Dependable Secur. Comput. (2024).

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

[43] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, Xiao Wang, Mystique: Efficient conversions for {zero-knowledge} proofs with applications to machine learning, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 501–518.

[44] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, Shumo Chu, ZEN: An optimizing compiler for verifiable, zero-knowledge neural network inferences, Cryptol. EPrint Arch. (2021).

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

[46] Zcash Foundation, Halo 2 documentation, 2024, (Accessed 14 November 2024), https://zcash.github.io/halo2/.

[47] Binbin Gu, Faisal Nawab, Zk-oracle: Trusted off-chain compute and storage for decentralized applications, Distrib. Parallel Databases (2024) 1–24.

[48] Scott Dykstra, PhD Jay White, Nate Holiday, Catherine Daly, David Alves, Ian Joiner, Space and time: The verifiable compute layer for Web3, 2024, URL https://assets-global.website-files.com/642d91209f1e772d3740afa0/658edf3cf26933c4878ec965_whitepaper.pdf, v1.0.

[49] Diogo V. Carvalho, Eduardo M. Pereira, Jaime S. Cardoso, Machine learning interpretability: A survey on methods and metrics, Electronics 8 (8) (2019) 832.

[50] Usha Bhalla, Suraj Srinivas, Himabindu Lakkaraju, Verifiable feature attributions: A bridge between post hoc explainability and inherent interpretability, in: ICML 3rd Workshop on Interpretable Machine Learning in Healthcare, IMLH, 2023.

[51] Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin, Nothing else matters: Model-agnostic explanations by identifying prediction invariance, 2016, arXiv preprint arXiv:1611.05817.

[52] Scott Lundberg, A unified approach to interpreting model predictions, 2017, arXiv preprint arXiv:1705.07874.

[53] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, Yong Rui, Visualizing and comparing convolutional neural networks, 2014, arXiv preprint arXiv:1412.6631.

[54] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[55] Marco Esposito, Github repository, 2024, https://github.com/gufett0/xAI-oracle.

[56] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, Siamese Neural Networks for One-shot Image Recognition.

[57] dcbuilder.eth and Worldcoin Team, An introduction to zero-knowledge machine learning (zkml), 2023, (Accessed 21 October 2023), https://worldcoin.org/blog/engineering/intro-to-zkml.

[58] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, Eran Tromer, The hunting of the SNARK, J. Cryptology (ISSN: 1432-1378) 30 (4) (2017) 989–1066, http://dx.doi.org/10.1007/s00145-016-9241-9.

[59] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, Xia Hu, Score-CAM: Score-weighted visual explanations for convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 24–25.

[60] EZKL documentation, 2023, (Accessed 05 November 2023), https://docs.ezkl.xyz/.

[61] Bianca-Mihaela Ganescu, Jonathan Passerat-Palmbach, Trust the process: Zero-knowledge machine learning to enhance trust in generative ai interactions, 2024, arXiv preprint arXiv:2402.06414.

[62] Daniel Kang, Tatsunori Hashimoto, Ion Stoica, Yi Sun, Scaling up trustless DNN inference with zero-knowledge proofs, 2022, arXiv preprint arXiv:2210.08674.

[63] Jiasi Weng, Jian Weng, Gui Tang, Anjia Yang, Ming Li, Jia-Nan Liu, Pvcnn: Privacy-preserving and verifiable convolutional neural network testing, IEEE Trans. Inf. Forensics Secur. 18 (2023) 2218–2233.

[64] Haochen Sun, Tonghe Bai, Jason Li, Hongyang Zhang, Zkdl: Efficient zero-knowledge proofs of deep learning training, Cryptol. EPrint Arch. (2023).

[65] Suppakit Waiwitlikhit, Ion Stoica, Yi Sun, Tatsunori Hashimoto, Daniel Kang, Trustless audits without revealing data or models, 2024, arXiv preprint arXiv:2404.04500.

[66] Kasra Abbaszadeh, Christodoulos Pappas, Dimitrios Papadopoulos, Jonathan Katz, Zero-knowledge proofs of training for deep neural networks, Cryptol. EPrint Arch. (2024).

[67] Thomas Krendl Gilbert, Megan Welle Brozek, Andrew Brozek, Beyond bias and compliance: towards individual agency and plurality of ethics in AI, 2023, arXiv preprint arXiv:2302.12149.