Proof of Attribution: Powering Explainable and Payable AI on OpenLedger

Openledger

June 2025

Abstract

This paper introduces *Proof of Attribution*, the foundational mechanism behind **OpenLedger**, an AI blockchain where data, models, and intelligent agents evolve onchain. OpenLedger unlocks liquidity across the AI stack by enabling transparent and verifiable attribution of data influence in model inference. We propose a dual-method system: influence function approximations for small models and suffix-array-based token attribution for large models. At the core are *DataNets*, where data contributors collaborate to build high-quality, specialized datasets designed for LLM-ready augmented intelligence. Models log their training provenance, enabling deterministic attribution, explainability of models, and real-time reward distribution. This turns data into an economic asset and builds a transparent, incentive-aligned AI ecosystem.

1 Introduction

Artificial Intelligence systems are increasingly built on massive collections of data, often aggregated from the open internet or contributed by individuals and communities. These datasets underpin the capabilities of models across domains including language, vision, biology, and robotics. Yet despite the central role of data in AI performance, there is no widely adopted mechanism to recognize or reward the original data contributors. As a result, contributors remain disconnected from the value their data helps generate, and the incentives to share high-quality, domain-specific data are weak.

This paper introduces Proof of Attribution, a technical and economic framework designed to address this gap. It establishes a verifiable link between model behavior and the training data that influenced it, and uses this link to calculate and distribute contributor rewards. The goal is to make training data a first-class, onchain asset that participates in the value creation process of machine learning.

At the core of this framework is the ability to trace which parts of a model's output can be mapped to specific pieces of training data. This attribution must be both precise and scalable, working across model sizes and modalities. For small, specialized models, influence is computed using gradient-based methods that approximate how the removal of a data point would affect loss on a given prediction. For large language models, attribution is derived by comparing output tokens against a compressed representation of the training corpus to identify memorized spans. In both cases, the influence scores are used to assign credit to datasets and contributors, forming the basis of an inference-level reward system.

The system is built on a new onchain primitive called the DataNet. Each DataNet represents a structured dataset contributed by one or more users, recorded with associated metadata, timestamps. Models log their training provenance using these DataNets, which allows for deterministic tracking of which datasets contributed to a given model version. This design enables transparent attribution and reward distribution at the level of individual inferences.

The architecture outlined in this paper enables a new class of decentralized AI infrastructure. One where contributors are economically aligned with model performance, and attribution is enforced at the protocol level. By shifting incentives toward data quality, traceability, and reuse, Proof of Attribution lays the groundwork for a more equitable, transparent, and composable AI ecosystem.

2 The Need for an Onchain Attribution Framework

As artificial intelligence continues to grow in scale and impact, it brings with it a heightened expectation for accountability, transparency, and fairness throughout the development pipeline. At the center of this transformation is data, the raw material that defines a model's capabilities and limitations. Despite its foundational role, training data is often treated as an anonymous and static input. The individuals and institutions responsible for producing or curating that data are rarely acknowledged or compensated.

This problem is both economic and structural. Without a framework that attributes model behavior to the specific data that shaped it, contributors are excluded from the value they helped generate. Auditors and researchers are unable to trace decisions back to original data sources. Builders cannot ensure that licensing conditions are honored. What is missing is a system that links model outputs to their training data with cryptographic verifiability, and that operates as part of the core machine learning lifecycle.

An onchain attribution framework addresses this need by encoding ownership, contribution, and influence as programmable, stateful objects. This approach enables the transparent tracking of data across training and inference phases, and provides a foundation for new economic models that compensate contributors based on measurable impact. The rest of this section outlines the three main drivers behind such a system: explainability, monetization of influence, and traceability.

2.1 Explainability

Understanding why a model behaves the way it does often requires insight into the data that influenced it. While many interpretability tools examine a model's internal mechanisms, few reveal the origin of its behavior in terms of specific training examples. Attribution at the data level makes this possible by identifying which inputs led to a given output.

For example, in a medical setting, it becomes possible to ask whether a diagnosis was influenced by real clinical cases in the training set. In education, one can identify whether an answer mimics a particular source. This kind of token-level provenance gives users, developers, and auditors a concrete basis for trust and validation. It also helps detect overfitting, leakage, or spurious correlations in models by exposing when memorized content is responsible for predictions.

2.2 Influence and Monetizing Data Contributors

Current AI pipelines collect data from users, scrapers, forums, and volunteers. These contributors are rarely acknowledged, and never rewarded based on the performance or usage of the models trained on their data. Attribution enables a new model where contributors can earn rewards not just for providing data, but for the actual influence their data has on model behavior.

This idea transforms data into a dynamic asset. It is no longer a static file that disappears into a training run. Instead, it becomes part of a long-term value chain. When a model generates an output and the system can prove that specific data points contributed to that output, the associated contributors can be compensated in real time. This creates a market for high-quality, well-curated, and diverse data, and encourages domain experts to contribute in return for measurable rewards.

2.3 Transparency and Traceability

Today, there is no clear record of what data was used to train most models. There are no public logs, no audit trails, and no consistent metadata. This lack of transparency makes it difficult to reproduce results, enforce licenses, or ensure ethical boundaries are respected. It also raises the risk of misuse, whether intentional or not.

An onchain framework solves this by logging every step of the data lifecycle on a public ledger. Each dataset is registered with a unique identifier. Each training run records which datasets it used, when it was trained, and how the data was partitioned or weighted. Each inference event can then be tied back to these records. This structure allows developers, contributors, and even regulators to trace model behavior to its source, check whether attribution is being handled correctly, and confirm that rewards are distributed as promised.

3 Understanding DataNets and the DataNet Registry

At the heart of the Proof of Attribution framework is the concept of the DataNet, a modular, onchain dataset created through community contribution. Each DataNet represents a focused slice of knowledge tailored for a specific domain or task. Rather than collecting generalized data, DataNets are built for well-defined niches such as legal contracts, code snippets, medical transcripts, sensor streams, or fine-grained question-answer pairs. This focused structure enables high attribution precision and relevance.

DataNets are open to anyone in the community to create, contribute to, or curate. When a contributor uploads data to a DataNet, they are not simply adding content. They are becoming part of a verifiable, composable dataset that can later be traced during model training and inference. Each contribution is recorded with metadata such as contributor identity, upload timestamp, license terms, preprocessing status, and optional quality scores. Contributors may also stake tokens to increase the visibility and economic weight of their contributions.

This process turns DataNets into live, community-curated datasets. Over time, as more contributors upload data and as more models train on the content, each DataNet evolves into a high-signal, niche corpus backed by transparent provenance. It becomes more than a static file. It becomes an economic object that generates attribution-based rewards based on how much influence it has on downstream models.

All DataNets are indexed in a global registry called the DataNet Registry. This registry tracks the following:

- **Dataset Identifier:** A unique content-based hash representing the DataNet.
- **Contributor Records:** A complete list of contributors and their relative stake or data volume.
- **Usage Logs:** A record of which models used the DataNet and how frequently.
- **Attribution Records:** A ledger of influence scores derived from inference-level traces.

The registry is public and queryable. Model developers can inspect which DataNets a given model was trained on, along with attribution weights and historical influence. Contributors can verify whether and how their data is being used. Communities can build reputation systems around DataNet quality, originality, and impact.

Registering a Datapoint

When a datapoint is submitted to a DataNet, the following sequence occurs:

1. The contributor chooses an existing DataNet or deploys a new one aligned with the domain of the data.
2. The datapoint is submitted through a signed transaction including content and metadata.
3. The datapoint is assigned a deterministic hash to ensure deduplication and enable traceability.
4. The content is stored offchain and made available through DA layer, while the metadata and hash are committed to the DataNet contract onchain.
5. The DataNet updates its internal index, which tracks all datapoints, contributors, and their cumulative influence score over time.

This transforms raw content into an attribution-ready record. Once indexed, the datapoint becomes queryable, traceable during inference, and eligible for attribution-based rewards.

Example: Registering a Medical Snippet

Suppose Maya, a healthcare researcher, wants to contribute to a DataNet focused on ophthalmology. She submits a data snippet:

> *"Red-green color blindness is typically inherited in an X-linked recessive pattern and affects the cone cells in the retina."*

Maya includes metadata tagging the snippet as a 'clinical-fact' . The system hashes the normalized text, stores the raw content, and records the hash and metadata in the DataNet smart contract. This datapoint now has a permanent onchain fingerprint and is indexed in the DataNet registry.

Later, a healthcare assistant model trained partially on this DataNet responds to a user asking about color blindness. The attribution engine matches Maya's contribution during inference, identifies its influence on the output, and triggers an onchain reward flow to her wallet, along with a verifiable attribution record.

DataNets are permissionless by design. Anyone can deploy a new one, aligned with a domain or idea they care about. However, a decentralized validation layer helps ensure consistency, relevance, and resistance to duplication or spam. Communities can collectively curate, upvote, or challenge contributions, creating a high-trust environment for data collaboration.

In this system, contributors are not just passive uploaders. They are stakeholders in an ongoing economic loop. If a contributor adds data that turns out to be influential for model predictions, they earn attribution rewards not just once, but every time their data contributes to an output. By aligning incentives around influence rather than volume, the DataNet structure encourages quality over quantity and creates a sustainable, collaborative ecosystem for building high-quality training datasets.

4 Finding Data Influence for Smaller Specialized Models

Modern AI systems depend on large volumes of training data. For community-driven models, understanding how each data point affects the model's predictions is crucial for ensuring transparency, supporting auditing, and enabling contributor rewards. For smaller models, particularly those fine-tuned using techniques like Low-Rank Adaptation (LoRA), we can take advantage of their limited parameter space to compute attribution in a tractable manner.

4.1 Motivation

Traditional influence function estimation methods rely on computing or approximating the inverse of the Hessian matrix of the training loss. This becomes computationally expensive and memory intensive as model size increases. Even iterative methods such as LiSSA suffer from instability and large resource requirements. Therefore, we seek an approach that retains theoretical rigor while introducing an efficient approximation that supports real-time inference attribution.

4.2 Closed-Form Influence Approximation

Let us denote the influence of a training data point (x_i, y_i) on a test point (x_k, y_k) using a layer-wise decomposition across L layers of a neural network. The standard influence function involves the term:

$$I(x_k, y_k) = -\sum_l v_l^T H_l^{-1} \nabla_{\theta_l} \ell(x_k, y_k)$$

where H_l is the Hessian at layer l, and v_l is a directional derivative that depends on gradients with respect to training samples.

We approximate H_l^{-1} by reordering the matrix inverse and expectation. Instead of computing the inverse of the average Hessian, we average the inverse of individual per-example approximations:

$$\frac{1}{n} \sum_n [\nabla_{\theta_l} \ell_i \nabla_{\theta_l} \ell_i^T + \lambda_l I]^{-1}$$

This approximation is efficient because the inverse of a rank-one update to a diagonal matrix can be computed in closed form using the Sherman–Morrison formula. The approximate inverse becomes:

$$\frac{1}{\lambda_l}\left[I - \frac{\nabla_{\theta_l}\ell_i \nabla_{\theta_l}\ell_i^T}{\lambda_l + \nabla_{\theta_l}\ell_i^T \nabla_{\theta_l}\ell_i}\right]$$

Substituting this into the influence function and aggregating across layers, the final closed-form approximation becomes:

$$\hat{I}(x_k, y_k) = \sum_l \frac{1}{\lambda_l} \cdot \frac{1}{n} \sum_n \left[\frac{L_{li}}{\lambda_l + L_{lii}} \cdot (L_{lik} - L_{lk})\right]$$

where $L_{li} = v_l^T \nabla_{\theta_l}\ell_i$ and $L_{lij} = \nabla_{\theta_l}\ell_i^T \nabla_{\theta_l}\ell_j$. This closed-form expression significantly reduces compute time while maintaining useful granularity over layer-wise attribution.

4.3 Error Bound and Theoretical Guarantees

Although the two operations — averaging before inversion and inverting before averaging — are not mathematically equal, the spectral norm of their difference is bounded. Let us define:

$$S_{li} = \nabla_{\theta_l}\ell_i \nabla_{\theta_l}\ell_i^T + \lambda_l I$$

We denote the error between the original and approximated inverse as:

$$\left\| \frac{1}{n}\sum S_{li}^{-1} - \left[\frac{1}{n}\sum S_{li}\right]^{-1} \right\|_2$$

It can be shown that this error scales with $O(d_l^2)$, where d_l is the dimension of parameters at layer l. In practice, for low-rank adapted models where d_l is small, this approximation yields accurate results with a negligible trade-off in fidelity.

4.4 Empirical Evaluation

To evaluate the effectiveness of the closed-form influence method, we use models such as RoBERTa and LLaMA 2 fine-tuned with LoRA on datasets including GLUE and synthetic noisy datasets. We examine three use cases:

- **Approximation Fidelity:** We compare influence scores with exact Hessian-based scores and calculate Pearson correlation coefficients across datasets. Our method consistently achieves higher correlation than Hessian-free and LiSSA-based estimates, with correlation dropping gracefully as LoRA rank increases.
- **Mislabeled Data Detection:** We test how well influence scores can identify incorrectly labeled training samples. Influence scores computed via our approximation achieve significantly higher AUC than LiSSA and Hessian-free methods, and often match or exceed exact scores.
- **Identification of Influential Points:** Using tasks such as sentence transformation, math problem reasoning, and image generation, we measure whether influential examples in the training set align with test output behavior. The method shows high AUC and recall in class-detection tasks, outperforming other methods across both LLMs and diffusion models.

4.5 Performance Metrics

The proposed method runs significantly faster and with lower memory consumption than traditional alternatives. For instance, computing influence scores for 4500 points on GLUE-QQP takes approximately 13 seconds, compared to 70 seconds for LiSSA and over 11,000 seconds for exact inversion. Memory usage scales with $O(D)$, as opposed to $O(D^2)$ in matrix-based approaches.

The approximation method offers a pragmatic balance between accuracy, efficiency, and scalability, making it ideal for attribution in community-trained, fine-tuned AI models deployed on OpenLedger.

5 Why OpenLedger Chooses Infini-gram for Attribution in Large Models

As the scale of language models and training datasets continues to increase, traditional attribution mechanisms fail to meet the requirements of efficiency, precision, and interpretability. OpenLedger adopts *Infini-gram* as the core method for data influence estimation in medium and large specialized language models due to its symbolic, scalable, and auditable structure.

This section outlines the limitations of previous methods, reviews classical n-gram models, and introduces Infini-gram as a suffix-array-based ∞-gram attribution framework, explaining how it solves key bottlenecks in attributing training data influence to model outputs.

5.1 Limitations of Gradient-Based Attribution

Most existing approaches to data attribution in language models rely on gradient-based methods such as influence functions or representational similarity metrics. While these methods are tractable for smaller, fine-tuned models, they fail to scale to medium or large specialized language models operating over trillion-token corpora. Specifically:

- **Storage and Memory Constraints:** Influence computation for each data point requires access to gradient vectors and possibly Hessian approximations, which are infeasible to store and compute at scale.
- **Opaque Representations:** Embedding-based similarity metrics provide limited interpretability and often capture high-level semantics instead of exact spans or token-level overlap.
- **Model Access Requirements:** These methods require backpropagation or access to model internals, which is not viable in cases where only black-box access is available (e.g., via inference APIs).
- **Insufficient Context Fidelity:** Traditional attribution fails to preserve token-level fidelity, particularly for long-form documents where a specific phrase—not the general topic—triggers the output.

As such, existing attribution techniques are ill-suited for tracing token-level influence in medium or large specialized language models.

5.2 Classical N-gram Language Models for Attribution

Before the emergence of neural architectures, n-gram models were the de facto standard for language modeling and attribution. An n-gram is a contiguous sequence of n tokens, and its probability is computed based on frequency counts:

$$P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \frac{cnt(w_{i-n+1}, \ldots, w_i)}{cnt(w_{i-n+1}, \ldots, w_{i-1})}$$

Where:

- w_i is the target token,
- $w_{i-n+1}, \ldots, w_{i-1}$ is the context window,
- $cnt(\cdot)$ is the number of times the sequence occurs in the corpus.

These models were transparent and deterministic, making them naturally suited for attribution. However, they faced critical challenges:

- **Fixed Context Size:** Typically limited to $n \leq 5$, restricting contextual expressivity.
- **Sparsity:** Many meaningful token sequences do not appear in even large datasets.

- **Storage Complexity:** High-order n-gram tables grow exponentially with vocabulary size and context length.
- **Literal Matching Only:** No support for generalization or paraphrase handling.

Smoothing techniques like Katz and Kneser-Ney partially alleviated sparsity, but these methods still lacked scalability and flexibility for modern use cases.

5.3 Infini-gram: Scalable Token-Level Attribution via Unbounded N-grams

To address these issues, OpenLedger adopts *Infini-gram*—a nonparametric attribution framework that replaces fixed-window n-grams with dynamically selected, longest-match suffixes. It enables precise, real-time, and transparent attribution for trillion-token corpora without requiring model access.

The core idea is to compute the ∞-gram probability using the longest available context from the training corpus:

$$P_\infty(w_i \mid w_{1:i-1}) = \frac{cnt(w_{i-n+1:i})}{cnt(w_{i-n+1:i-1})}$$

Where:

$$n = \max\{n' \in [1, i] \mid cnt(w_{i-n'+1:i-1}) > 0\}$$

This formulation ensures that predictions are attributed to the most specific matching context available, falling back only when required.

Sparse vs Dense Attribution

- **Sparse Attribution:** If a unique, long context appears only once in the dataset, the attribution is nearly deterministic.
- **Dense Attribution:** For generic or ambiguous contexts, the output reflects a probabilistic distribution over multiple plausible next tokens.

This context-sensitive behavior allows fine-grained attribution even in the presence of ambiguity.

5.4 Indexing Architecture and Runtime

Infini-gram leverages a suffix-array-based indexing mechanism optimized for fast lookup and large-scale storage.

Suffix Array Construction

- All tokens are concatenated into a single sequence.
- The suffix array lexicographically sorts all suffixes of this sequence.
- This supports efficient longest-prefix search in $O(\log N)$ time.

Storage Format

- **Token Array:** 2 bytes/token
- **Suffix Array:** 5 bytes/token
- **Total Index:** $\sim$7 bytes/token $\rightarrow$ $\sim$35 TB for a 5T-token corpus

Query Execution Pipeline

1. Locate context via binary search on the suffix array.
2. Compute frequency of both prefix and full n-gram.
3. Estimate probability as a relative frequency.
4. Apply recursive backoff if no match is found.

Performance Benchmarks

Query Type	Latency
Count ($n = 5$)	∼20 ms
∞-gram Probability	∼135 ms
Full Distribution	∼180 ms

5.5 Attribution-Centric Use Cases

Infini-gram enables attribution capabilities across a wide spectrum of applications relevant to OpenLedger's mission:

- **Token-Level Traceability:** Outputs can be directly linked to exact spans in the training data, preserving provenance and auditability.
- **Memorization Analysis:** High-confidence matches expose potential memorization in specialized models.
- **Data Contamination Detection:** Evaluation corpora can be compared to training indices to surface unintentional leakage.
- **Dataset Optimization:** Repeatedly triggered or redundant spans can be identified and pruned from training data.
- **Regulatory Compliance:** Allows provable disclosure of training data origin for enterprise and public sector applications.
- **Reward Distribution for Data Contributors:** Span-level attribution enables proportional and verifiable reward sharing among data contributors, making it possible to incentivize high-quality contributions and enforce fair data usage in collaborative AI systems.

5.6 Integration with Neural Models and Hybrid Attribution

While Infini-gram provides deterministic and interpretable attribution, it is not intended to replace neural models in generation or reasoning tasks. Instead, it serves as a complementary system that enhances transparency and precision in attribution pipelines.

Hybrid Probability Interpolation

OpenLedger employs Infini-gram alongside medium and large specialized neural models through a hybrid interpolation strategy:

$$P(y \mid x) = \lambda \cdot P_\infty(y \mid x) + (1 - \lambda) \cdot P_{neural}(y \mid x)$$

Where:

- P_∞ is the Infini-gram symbolic estimate,
- P_{neural} is the probability from the neural model,

- λ is a context-aware interpolation weight.

For high-confidence spans (i.e., sparse, uniquely matched n-grams), λ is increased to give more weight to Infini-gram. In ambiguous or low-confidence contexts, the neural model takes precedence.

By replacing gradient-based attribution with ∞-gram symbolic tracing, OpenLedger introduces a transparent, reproducible, and scalable system for measuring the influence of data in modern language models. The combination of precise matching and real-time performance positions Infini-gram as a foundational tool for building accountable and rewardable AI systems on-chain.

6 Attribution Aggregation and Reward Distribution

Once token-level or gradient-level influence scores are computed, either through span matching (Section 5) or influence functions (Section 4), the next step is to elevate these attributions to the level of DataNets. This enables contributor rewards, governance structures, and analytics to operate at the unit of community curation rather than individual samples.

Each training sample is already mapped to a DataNet, which is a structured onchain container of semantically aligned data. During aggregation:

- For every inference output z_t, extract the set of training points $\{z_1, z_2, \ldots, z_n\}$ that influenced it.
- Each training point z_k is mapped to a DataNet D_i using registry hashes that record its origin, stake, and contributor share.
- The total influence of D_i is computed by summing the influence values of all samples from D_i.
- These scores are normalized across all contributing DataNets to yield a proportional distribution.

Formally:

$$I(D_i, z_t) = \sum I(z_k, z_t), \quad forall z_k \in D_i$$

$$W(D_i, z_t) = \frac{I(D_i, z_t)}{\sum I(D_j, z_t)} \quad forall j$$

These weights, $W(D_i, z_t)$, become the basis for reward distribution and are recorded onchain along with model identifiers, output hashes, and attribution metadata. This builds a complete, tamper-proof trail from training data to output.

7 Reward Attribution Flow at Inference

The purpose of attribution is to allow contributors to be rewarded each time their data influences a prediction.

When a user submits an inference request, the model generates an output z_t. This output is not random—it is shaped by training data registered through DataNets. Attribution algorithms such as gradient-based sensitivity analysis or -gram span matching are applied post-inference to identify the datapoints most responsible for the result.

Every datapoint is indexed and registered onchain via the DataNets registry. This allows the system to establish Proof of Attribution (PoA) by tracing each token in the output back to its origin. The output, along with model metadata and timestamp, is then committed onchain—making it publicly auditable and reproducible. With this traceability in place, we can deterministically identify and reward contributors whose data shaped the model's behavior.

To illustrate this : imagine a healthcare assistant model that answers a user's query about "what causes red-green color blindness." The model responds with:

"Color blindness primarily results from inherited genetic defects affecting photoreceptors in the retina. These photoreceptors, known as cones, are responsible for color vision. The most common form is red-green color blindness, usually inherited in an X-linked recessive pattern. A lack of certain photopigments in these

cones leads to difficulty distinguishing between certain colors. It affects males more frequently than females due to the X chromosome inheritance pattern."

Using PoA, the attribution engine identifies the exact datapoints that influenced this output. In this case, the model draws from three sources:

- `ui-intelligence/ui-train-0017.json.gz` — A design guideline document discussing color accessibility, contributed by **Henry**, which highlights red-green distinctions and their impact on UI design.
- `color-theory/ct-0062.json.gz` — A corpus on color theory and accessibility standards, contributed by **Bailey**, offering foundational knowledge on red-green color blindness and user experience best practices.
- `healthwiki/genetic-vision-0009.json.gz` — A genetics and ophthalmology dataset, contributed by **Maya**, detailing the X-linked inheritance pattern associated with vision disorders.

Each of these contributors is registered through DataNets. The model's attribution layer computes their relative influence on the output, and a share of the inference fee is distributed accordingly. All of this happens onchain, with a transparent proof of attribution and reward flow tied to each response.

- A user initiates an inference request. The model processes the prompt and returns an output z_t.
- The output, model metadata, and timestamp are committed onchain for transparency.
- Attribution methods are applied post-inference, generating influence scores based on either gradient sensitivity or token match.
- The influence is grouped and normalized across all participating DataNets.

The inference fee F is split into components:

$$F = F_{platform} + F_{model} + F_{stakers} + F_{contributors}$$

The contributors' share is calculated as:

$$F_{contributors} = \delta \times (F - F_{platform})$$

$$Reward(D_i, z_t) = F_{contributors} \times W(D_i, z_t)$$

Here, δ is the allocation ratio set by protocol governance, and $W(D_i, z_t)$ is the DataNet's influence share. Every payout includes proof of attribution, allowing contributors to verify influence and payment history.

8 Advanced Extensions

The Proof of Attribution framework enables new primitives that extend attribution into modular training pipelines, long-term contributor dynamics, and verifiable adapter ecosystems. These extensions convert attribution from a passive audit trail into an active layer of infrastructure for decentralized AI development.

8.1 Adapter-Level Attribution

Modern language model training frequently uses adapters, such as LoRA or QLoRA, for efficient finetuning. Each adapter is linked to a specific dataset and training event. During inference, attribution is decomposed across the base model and the adapter. The reward mechanism reflects this split. Adapter-specific influence is linked to the underlying DataNet, and contributors who produced the data powering the adapter receive influence-weighted rewards whenever that adapter is used in inference.

8.2 Fine-Grained Metadata Attribution

DataNets support rich metadata tagging at the sample level, including domain labels, language, license type, and data quality. Attribution scores can be further decomposed across metadata attributes. This enables auditing for bias, domain-specific incentive programs, or policy constraints, such as licensing rules. For example, a legal chatbot may show that 65 percent of its inference influence came from English-language, open-license, jurisdiction-tagged case law data.

8.3 Attribution in Governance and Curation

Attribution is not only a mechanism for reward. It can also power governance rights. DataNets with high influence across multiple production models can be granted higher voting power within the protocol. Curation of new datasets, adapter prioritization, and fee distribution rules can all be weighted by past influence.

8.4 Attribution Graphs and Leaderboards

All influence weights, model-data relations, and inference events are stored as part of a public attribution graph. This graph enables real-time analytics for contributor reputation, dataset saturation, and underutilized niches. Leaderboards can rank the most influential DataNets, most used adapters, and most rewarded contributors per model family.

9 Conclusion

Proof of Attribution introduces a principled framework for data-centric AI that rewards contributors based on actual influence rather than assumptions about data value. It supports both gradient-based and substring-based attribution pipelines, enabling application across compact specialized models and frontier-scale language models.

This system enables modular reward attribution across datasets, adapters, and training contexts. Contributors maintain visibility and economic rights to their data, even across recursive reuse or synthetic generation. Combined with onchain registration and auditability, Proof of Attribution transforms data into a transparent, composable, and rewarding digital asset.

The framework redefines what it means to own, track, and be paid for the impact of data in a decentralized AI ecosystem.

References

- Tworkowski, M., Wu, C., Jung, D., Li, X. L., & Zitnick, C. L. (2024). Infinigram: Scaling Attribution for Large Language Models. arXiv:2401.17377v4.
- Kwon, Y., Wu, E., Wu, K., & Zou, J. (2024). DataInf: Efficiently Estimating Data Influence in LoRA-tuned LLMs and Diffusion Models. arXiv:2310.00902.
- Koh, P. W., & Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. In ICML.
- Basu, S., Chopra, A., Zhou, P., & Henderson, K. (2021). Influence Functions in Deep Learning are Fragile. In NeurIPS.
- Pruthi, G., Liu, F., Gardner, M., & Neubig, G. (2020). Estimating Training Data Influence by Tracing Gradient Descent. In NeurIPS.
- Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., & Madry, A. (2022). Datamodels: Predicting Predictions from Training Data. In ICML.
- Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., et al. (2023). Extracting Training Data from Diffusion Models. In IEEE S&P.

- Tirumala, A., Gao, T., Lin, Z., Bansal, M., & Artzi, Y. (2022). Memorization in Fine-tuned Language Models. In ACL.
- Tramer, F., Zhang, F., Juels, A., Reiter, M. K., & Ristenpart, T. (2016). Stealing Machine Learning Models via Prediction APIs. In USENIX.
- Lee, K., Carlini, N., Jagielski, M., Tramer, F., & Song, D. (2022). Deduplicating Training Data Makes Language Models Better. In ACL.
- Henderson, P., Jojic, O., & Ghosh, D. (2023). Data Attribution in Foundation Models with TracIn. In NeurIPS.
- Liang, P., Potts, C., & Jurafsky, D. (2022). On the Measurement of Societal Biases in Texts. JAIR.
- Zhang, F., Juels, A., & Ristenpart, T. (2022). Extracting Training Data from Generative Language Models. In USENIX.