International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

DeTrust ETH: A Real-Time Fraud Intelligence Framework for Ethereum Using Temporal Graph Analysis and Explainable AI

Sarthak Kumar Singh and Navya Dhaka
Department of Computer Science and Engineering, Delhi Technological University, Delhi, India

Abstract- **Fraud detection on Ethereum is challenging because of the anonymity, speed and graph structure of blockchain transactions. While prior research has proven the effectiveness of using machine learning classifiers, Graph Neural Networks (GNNs) and behavioural heuristics to detect fraudulent transactions, most systems are offline and fail to consider real-world deployment challenges for real-time blockchain analytics. Here, we present DeTrust ETH, an operational fraud intelligence system for real-time tracking of Ethereum transactions on the Sepolia testnet.**

DeTrust ETH aims for the integration of five operational considerations: (1) real-time blockchain ingestion with Web3.py, (2) explainable machine learning with XGBoost and SHAP, (3) light-weight graph-based transaction tracing and risk propagation, (4) temporal trust decay and behavioural anomaly detection, and (5) tamper-resistant on-chain persistence of trust scores using Solidity smart contracts. The system maintains an in-memory directed transaction graph for real-time edge insertion and updating, circular-flow tracing, funding pattern tracing and fast path tracing, avoiding the retraining overhead of Graph Neural Networks (GNNs).

Our experimental results demonstrate the median graph-query time is less than 20 ms, the system can handle 222.82 requests per second with a concurrent load, and 93.44% fraud recall with a recall-favouring threshold. Moreover, benchmarking also demonstrates the trade-offs between light graph analytics and sophisticated GNNs. The novelty of this work lies in the design of a real-time, low-latency fraud intelligence architecture that integrates explainable machine learning, temporal trust modeling, and lightweight graph analytics under streaming blockchain constraints. Rather than focusing solely on predictive performance, the proposed system addresses practical deployment challenges such as continuous data ingestion, incremental graph updates, and interpretable decision-making. The code can be found at: DeTrust-ETH

Keywords- **Ethereum fraud, explainable AI, SHAP, graph analytics, transaction tracing, temporal trust modeling, blockchain security, XGBoost, Web3.py, real-time analytics**

I. INTRODUCTION

Native financial frauds on the Ethereum blockchain, such as phishing, wallet drainers, money launderers through mixers, rug pulls, wash trading and bot-driven scams, are emerging. Ethereum's anonymity and transparency allow criminals to divide the funds into a series of intermediary wallets. In general, fraud-detection studies on Ethereum fall into three broad categories: rule-based behavioural systems, supervised machine learning (ML) classifiers and graph-based deep learning models (such as Graph Neural Networks or GNNs).

Though they perform well in classifying fraud, most past systems are evaluated offline on transaction snapshots. Hence, they might not factor in deployment aspects such as real-time ingestion of blockchain data, real-time transaction fraud scores, explainable insights for human analysts, real-time graph updates, scalability, and verifiable persistence of fraud decisions. Our paper investigates the possibilities of deploying light graph analytics and explainable machine learning to produce valuable real-time fraud intelligence at the cost of GNN inference.

We present DeTrust ETH, a deployable fraud intelligence system that integrates real-time ingestion of Ethereum transactions, explainable XGBoost fraud score, in-memory graph tracing, temporal trust modelling, graph-based risk propagation, and on-chain persistence of trust decisions. Unlike prior research that mostly focuses on accuracy on historical data, DeTrust ETH focuses on real-time deployment.

A. Contributions

This paper makes three contributions. First, we deploy a real-time Ethereum fraud intelligence system that captures, enriches and scores transactions during the block confirmation period. Second, we use a lightweight temporal graph analytics module that facilitates real-time circular flow detection, funding tracing and limited risk management without retraining the graph. Third, we implement an explainable system based on SHAP, analyst summaries and temporal trust in an operational architecture that we test with load in the presence of real-time queries. Importantly, We integrate **explainable machine learning (SHAP), temporal trust modeling, and on-chain auditability** into a unified system, demonstrating the feasibility of interpretable and deployable fraud detection in streaming blockchain environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

II. RELATED WORK

A. Rule-Based Fraud Detection

The original work (Aparecium: Detecting Scam Behaviors on Ethereum, 2023 [1]) focuses on the techniques of quantifying the scam behaviour as converting the pattern of transactions into a quantitative measure. It identifies Ethereum scams via heuristics that relate to transaction behaviours such as spikes in transactions, wallet balance fluctuations and wallet volatility. These are transformed into the features like - abnormal value outflow, transaction spikes or abnormal drop in wallet balance. Our machine learning model uses these signals in combination to assign a scam-risk score or the likelihood of a transaction or wallet being related to a scam. This score enables early detection of scam like behaviour even in the presence of noisy data. DeTrust ETH extends some of the basic ideas of this research that are behaviour-centred, such as volatility analysis of balance, with graph and temporal reasoning.

B. Supervised Machine Learning

An earlier study by Shukla and Sharma [2] demonstrated that XGBoost outperforms Logistic Regression and Random Forest with Ethereum fraud data, because it can capture complex interactions between features, and handle class imbalance. This was later verified by other SciTePress fraud detection papers [3].

The paper Intelligent Fraud Detection in Ethereum Transactions Using Machine Learning[2] has shown that models like Random Forest and XGBoost can be used for identification of fraudulent transactions based on behavioural attributes, like transaction amount and volume. This article showed that traditional ML models can be used to detect the simple patterns in fake wallets and phishing sites. The study (Fraud Detection Using ML - SciTePress, 2024 [3]) compares various ML models using Ethereum transactions. It shows that: • Logistic Regression is not an effective one due to the fact that the patterns of blockchain fraud are very non-linear. • Random Forest is more effective because it can learn non-linear relationships, noisy features and reduces over-fitting with bagging. • XGBoost is more effective than random forest due to the fact that it learns aggressively interactions between the features, boosts and high-dimensional data of the blockchain. The authors have concluded that tree-based ensemble methods are the most appropriate to use for Ethereum fraud detection and XGBoost is the best performing in general. This is the reason why our project is heading in a direction of model selection and this is one reason why the concept of boosted trees is preferred by many of today's fraud- detection systems over classical systems.

DeTrust ETH enhances this work by incorporating engineered behavioural features, using a model explainability technique (SHAP) and embedding the model in a temporal graph-based pipeline. The machine learning (ML) component is not claimed to be an innovation, but rather the use of explainable ML in a real-time blockchain monitoring setting is emphasised.

C. Graph-Based Blockchain Analysis

Recently, many blockchain analyses have considered the wallet system as a graph, where wallets are the nodes and transactions are the edges [5][6]. GNN systems use these relationships to enhance fraud detection.

Amazon's researchers noted that "real-time inference of GNN models adds complexity to the implementation"[12]. In many GNN systems, batch retraining, storing the entire graph, expensive embeddings, inference time and high GPU memory are used. This can make online monitoring difficult to implement because of the ongoing edge insertions, and the requirement for fast processing. Instead of GNN embeddings, DeTrust ETH adopts a lightweight in-memory graph index for fast edge insertion, tracing transactions, cyclic detection, risk propagation with a limited budget (just enough to propagate risk through the tree of transactions), and pathing. This design improves the responsiveness and deployability (at the expense of offline benchmarking).

D. Explainable AI in Finance

SHAP is a widely used explainability framework in financial machine learning, due to the theoretically sound feature-attribution method [4]. While explainability has been used for other machine learning financial systems like credit scoring and anti-money-laundering, it has not been widely used in real-time Ethereum fraud monitoring. DeTrust ETH integrates SHAP explanations in analyst workstations, and generates natural language explanations.

E. Temporal Behaviour

Temporal patterns have emerged as a valuable characteristic in blockchain fraud detection as the wallets involved in fraud are known to display suspicious spikes in transactions, inactivity followed by sudden activity. Agarwal, Barve and Shukla [7] showed that malicious activity in permissionless blockchain networks is by nature temporal, as fraudulent wallets tend to have varying transaction patterns as time progresses, rather than being persistently suspicious. They demonstrated that temporal graph characteristics and constantly evolving wallet interactions are significant indicators of malicious behaviour in blockchain. Likewise, the Dynamic Trust Decay framework for blockchain oracles [8] h lighted that older interactions should decay in trust evaluation systems, as past interactions may not be a good indicator of future trustworthiness.

Building on their work, DeTrust ETH's temporal trust model not only accounts for accounts that have been inactive for a long time through exponential trust decay, but also for sudden spikes in the number of transactions and reactivation of accounts after a long period of inactivity. The proposed solution not only identifies temporal patterns, but also integrates temporal scoring into online real-time fraud detection.

F. Gap Analysis

The current state of the art is good on individual components such as behaviour heuristics, offline ML classification, graph representation learning, explainability. However, there are fewer systems that focus on the integration of real-time ingestion, fast graph updates, explainability, temporal trust adaptation, deployment. The DeTrust ETH project is focused on the deployment integration.

III. SYSTEM ARCHITECTURE

DeTrust ETH comprises five subsystems which are loosely connected so that we can ingest blockchain transactions in real-time and execute feature engineering and scoring with a ML model, then enrich the scoring with graph intelligence,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

temporal trust and persist this for visualisation. Batch asynchronous processing of transactions and real-time fraud detection is supported.

A. Real-Time Blockchain Ingestion

The oracle reads the Ethereum Sepolia network every few seconds using Web3.py. When blocks are finalised, the following operations are performed: transaction decoding, contract function decoding, data enrichment (enriching wallet details) and the creation of behavioural features. The enrichment data is via Etherscan APIs, Chainlink price feeds, wallet data and statistics. The enriched transaction is then passed to the fraud-analysis pipeline for scoring and graph analyses.

B. Explainable ML Risk Scoring

The fraud-scoring module uses XGBoost to score fraud. The features include balance volatility, per wallet transaction velocity, percentage of balance used, transaction fees, log features and non-linear combinations of these features. It returns a fraud probability which is converted into an operational risk score. For transparency, SHAP explanations are generated for each prediction. These are then used to generate analyst reports using a language-generation layer. The aim is not just to make a prediction, but also to provide a transparent service.

C. Lightweight Graph Intelligence Engine

1) Graph representation: The interactions among Ethereum wallets form a network as they transfer value between each other. DeTrust ETH does not investigate the transactions one by one, but examines the interactions in the form of a dynamic directed graph, stored in-memory for quick processing. The nodes represent Ethereum wallet addresses, and the directed edges represent transactions between the wallets. The directed edges are also used to store metadata about the transactions occurring between the wallets such as the amount transferred, the time of the transfer, the number of transfers and the most recent risk score.

This model is not like previous Graph Neural Network (GNN)-like models that are repeatedly retrained on large graph snapshots. This enables edges to be added and processed in a timely manner. This model has a lightweight, real-time monitoring framework to produce embeddings. This enables us to conduct light transactions and queries in real blockchain graphs.

2) Circular Flow Detection: One of the most common methods of laundering cryptocurrency is to move funds through "mixing accounts". Such laundering techniques can generate a cycle in the graph. In order to detect these cycles, the graph engine, when a new transaction is added to the graph, searches the graph with at most k-steps. In particular, its looking for cycles like:

$$A \rightarrow B \rightarrow C \rightarrow A$$

where money returns to an already seen wallet via one or more intermediary wallets. Such cyclical funding is often used to obfuscate the flow of funds (also known as layering) in money laundering schemes making it difficult to trace[10][11].

3) Funding Pattern Analysis: The system is not only interested in cycles but also in repeated fundings. Fraud systems typically use a wallet to repeatedly fund multiple other wallets downstream to create bot traffic, networks of mule-wallets or to dispense other funds stolen from victims before they are laundered.

The graph engine thus tracks recurring transactions and identifies wallets funding networks of incoming wallets over time. This may represent bot wallets or fraud schemes. The network will monitor the transaction count, the age of the wallets, the recurrence of the transactions and the downstream connections to detect suspicious funding transactions, although the transactions themselves may not be suspicious. The network perspective is important as many malicious wallets only exhibit a few behaviours. They are only seen as risky when connected to other wallets in the network.

4) Risk Propagation: Current transaction-based classifiers are stand-alone and may miss new wallets that are closely linked with known scam wallets. To mitigate this issue, DeTrust ETH presents a simple graph-based risk propagation mechanism which assumes that the fraud risk is a local property of the transaction graph.

The proposed risk propagation is inspired by the concepts of transaction tracing and laundering networks in blockchain forensics [9][10]. Rather than training GNNs to learn the embeddings in the graph, the proposed risk propagation is based on a deterministic formula and can be computed in real time. The risk score for a source wallet is:

$$R\_contrib = \alpha \times R\_source \times \gamma^{(d-1)} / (1 + d)$$

where α controls propagation strength, γ controls hop-based attenuation, and d represents graph distance within the transaction network. The formulation incorporates three key principles:

a) *Proportional influence:* Risk contribution is directly proportional to the source wallet's risk score.
b) *Exponential Hop Decay:* The term $\gamma^{(d-1)}$ ensures that risk influence diminishes exponentially with increasing graph distance.
c) *Path-Length Normalization:* The division by $(1+d)$ penalizes long transaction chains, reducing the influence of distant nodes and preventing over-propagation.

Theoretical Motivation of Risk Propagation. The formulation used for risk propagation draws on ideas from influence spreading and information flow in networked systems. In such settings, the effect of a node typically weakens as it moves further across the network, which is why an exponential decay term is used to reduce influence over multiple hops. At the same time, the normalization factor limits the contribution from long transaction paths, preventing risk from accumulating excessively through extended chains.

Similar approaches have been explored in trust propagation and network-based anomaly detection, where nearby connections are considered more relevant than distant ones.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

In this work, these ideas are adapted to suit real-time blockchain transaction graphs, allowing efficient computation while still capturing meaningful local risk relationships. This strategy allows timely detection of mule wallets and questionable intermediary wallets without spreading influence unduly throughout the network.

Crucially, the light-weight propagation strategy alleviates the retraining cost, memory consumption and inference time of deep graph-learning approaches, making it more feasible to be deployed in real-time fraud monitoring systems.

D. Temporal Trust Modelling

Blockchain fraud transactions can have diverse patterns. For instance, wallets can remain inactive for a long time and then be used for a fraud, phishing scam or laundering. Bot-driven attacks also involve large volume transactions within a short time period. Time independent trust scores thus fail to capture the dynamic nature of wallet activity.

Our approach uses a time dependent score that keeps updating the trust scores of wallets with respect to their recent activity. We use exponential decay to decay trust over time in between actions: S_decayed = S_prev × e^(−λΔt) S_prev is the trust at the previous time, λ is the decay coefficient and Δt is the time difference since the last action. This equation assumes that previous positive behaviour should not always lead to trust. As wallets go dormant, trust in their past behaviour wanes.

In addition to decay, the temporal model assigns behavioural penalties for unusual temporal patterns that have been observed in past criminal blockchain transactions [7][8]. These include sudden bursts in the number of transactions, very short transaction delays, and sudden awakening of dormant wallets. These are common in compromised wallets, botnet activities and the final working phases of scams. The system uses temporal reasoning in the real-time fraud detection pipeline to apply the notion of time to the concept of trust decay.

E. Data Storage and Dashboard

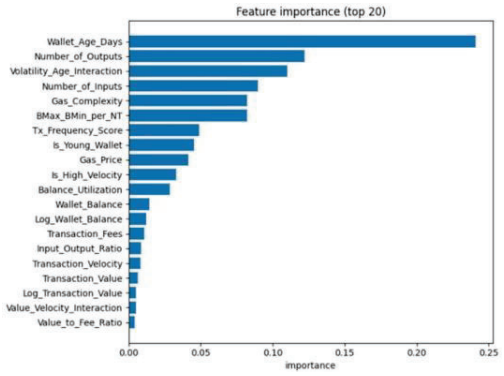
A Flask API is used as a communication server to receive transactions, update the graph, compute risk scores and temporal trust scores, and present them on the dashboard. The server architecture is designed for asynchronous processing, with timely feedback for monitoring. The system uses PostgreSQL for persistence of wallet state, transactions, graph edges, risk and temporal trust scores. The persistence framework is used for forensic analysis, historical replay and replay-based transaction analysis. The persistence framework also offers a real-time monitoring dashboard based on WebSocket. This dashboard is updated in real time, allowing analysts to see the live graph reorganisation, alerts, and risk-scores changes. For enhanced auditability, the trust scores are also written to an Ethereum smart contract written in Solidity. This creates a tamper-resistant trail of fraud risk scores, and prevents historical risk scores from being changed.

IV. EXPERIMENTAL SETUP

A. Datasets

We used the publicly available dataset of Cryptocurrency Scam Dataset in DQN Models that is available on Mendeley Data[13] for training and evaluation. We used this data because it has transaction data that is structured and it is present in real Ethereum transactions and is used in detecting frauds: transaction value, transaction fees, number of inputs, number of outputs etc. Moreover, the data size is very suitable as it is ready to use and labelled which makes it easier to train and evaluate. Even though we had trained our XGBoost model with the Kaggle data set[14], which does not contain most of the columns in the data set that were used in our machine learning model, our model is working quite well. This model has been trained with larger training data and identified the patterns of transactions which can be used to predict using very few features available in the Kaggle test data[14].

We tested the graph engine with real data feed and synthetic injection sequences of transactions similar to those described so far in the blockchain forensics literature [9][10][11], namely circular transactions, recycling of funds and mule-wallets.



We test the DeTrust ETH on the Ethereum Sepolia testnet as a simulated real-time production environment. The testnet enables real-time transaction ingestion, graph processing and inference, which are critical for reaching our research objectives. While testnet transactions may not replicate the intricacy and volume of Ethereum mainnet transactions, it allows us to test the system's responsiveness, data processing capacity, and near real-time fraud scoring. The testnet data also allows for synthetic injection of fraud patterns like circular fund transfers, layering transactions and transaction bursts, which are difficult to disentangle. This ensures that graph analysis and temporal reasoning modules of the system can be tested in a controlled environment.

B. Evaluation Objectives

The objectives of this evaluation are both predictive and performance. Beyond conventional studies that only assess classification performance, this work also studies latencies, graph update latencies, and API latencies under concurrent traffic.

C. Baseline Comparison

We compare the architecture of DeTrust ETH with other GNN designs on blockchain to illustrate the trade-offs. Specifically, the update (training) complexity, inference (prediction) efficiency, retraining requirements, interpretability, and practicality. The aim is not to prove that DeTrust ETH will always outperform the predictive accuracy of other GNNs, but to evaluate its practicality to real-time monitoring.

To evaluate the detection of graph-based fraud schemes such as circular fund flows, layering, and structured funding we synthetically inject sequences into the transaction stream. This is necessary because there is no ground truth for such

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

graph topologies in public blockchain data. Synthetic injection lets us control the structural properties (e.g., cycle length, fan-out and spread) and therefore deterministically evaluate the detection capacities. Importantly, this does not replace real-world testing, but rather, extends it to test for certain graph topologies frequently encountered in blockchain forensics. Controllable experiments are also adopted in other graph-based security research where ground truth is not available.

V. RESULTS

A. ML Performance

The sensitivity analysis was conducted on changing decision threshold between 0.01 and 0.9. Three metrics were tracked:

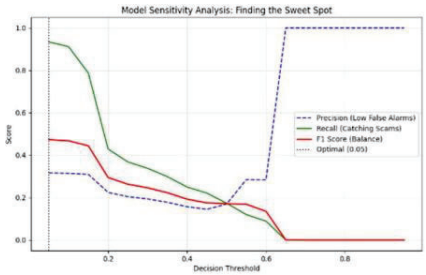
- Precision (capability to prevent false alarm)
- Recall (capability to detect stolen wallets)
- F1 Score (overall balance)

Recall is high at lower thresholds and precision improves at higher thresholds as is observed in the plot. The 0.25 cut-off was selected since it maintains extremely high recall and yet, at the same time, has acceptable precision in fraud-detection applications, in which false omission of a scam is significantly more expensive than false alarms on a normal wallet. With a threshold of 0.25 the model has been balanced enough to detect a scam and at the same time it has been made practical to be used in the real world.

For a recall-useful threshold of 0.25, the XGBoost classifier produced:

- Recall: 93.44%
- Precision: 31.68%
- Accuracy: 53.93%

We chose the threshold with a higher fraud capture at the cost of lower precision as is required in fraud monitoring systems.



Confusion Matrix (Threshold = 0.25) -

	Predicted Normal	Predicted Fraud
Actual Normal	3271 (TN)	4391 (FP)
Actual Fraud	143 (FN)	2036 (TP)

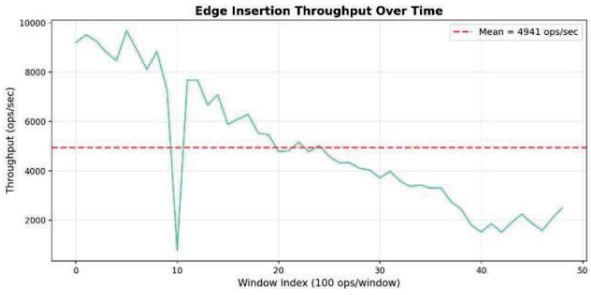
As it is shown in the confusion matrix, the system is able to reasonably detect the vast majority of fraudulent wallets at the cost of a greater number of false positives, which is a design choice to reduce risks in a sensitive setting. We are focusing on catching as many real scam transactions as possible and in the case of fraud detection, a false positive is higher than a false negative. The cost of a trade-off is well deserved, as missing a scam (FN) is much more detrimental than raising an alarm on a safe transaction (FP).

The high recall observed in our evaluation is also influenced in part by the characteristics of the Ethereum Sepolia testnet. In contrast to the mainnet, where transaction activity is highly diverse and often adversarial, the testnet environment typically involves experimental and low-risk interactions. This results in comparatively simpler transaction patterns, making anomalous or suspicious behavior easier to identify. It is also important to note that the model itself is trained on real-world Ethereum datasets. Therefore, while the evaluation environment may be less complex, the model is designed to generalize to more realistic and challenging scenarios. In a production setting on the mainnet, performance is expected to remain strong, although variations may occur due to the increased diversity and sophistication of transaction behavior.

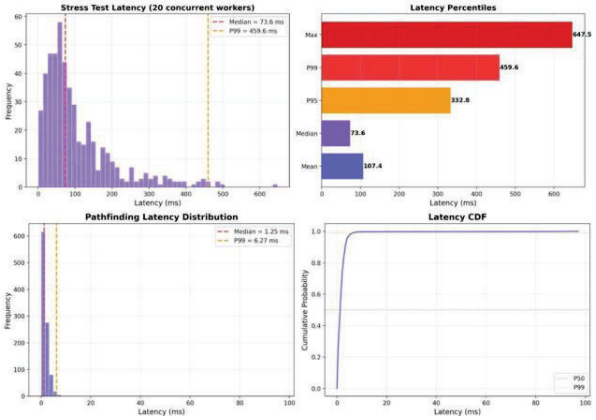
The slightly lower precision value in our evaluation (31.68%) is mainly due to two reasons: (i) deployment on the Ethereum Sepolia testnet, and (ii) a focus on high recall in the context of detecting fraudulent transactions in real-time. The Sepolia testnet comprises a combination of synthetic, experimental, and poorly labelled transaction patterns, which may not exhibit clear fraudulent traits. This results in a more blurred feature space and more false positives. In terms of system design, DeTrust ETH is designed to prioritise recall (93.44%), as to maximise the detection of potentially fraudulent transactions in the early stages. In practice, false negatives (missed fraud) are much more costly than false positives (false alarms) in fraud monitoring systems, and the latter can be further investigated by analysts. As a result, the observed trade-off in precision and recall is a deliberate operational choice in line with industry best practices in high-stakes settings, such as anti-money laundering (AML) and blockchain forensics.

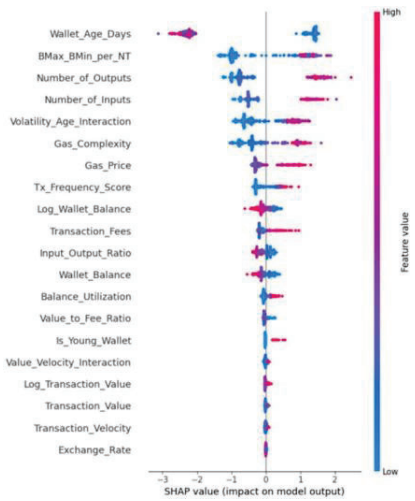
B. Graph Performance

We assessed the graph engine on a synthetic Ethereum wallet-to-wallet transaction graph with around 2,000 wallets and 10,000 transactions. The experimental results of the graph engine achieved a median path-finding latency of 1.25 ms (p99 = 6.27 ms), which is well below the set threshold of 20 ms. Circular-flow detection and repeated-funding pattern detection took less than 0.2 ms, showing the capability to detect fraud in real time. The median latency of dynamic edge insertion was 0.193 ms (or 4,941 operations per second). An experiment combining 500 simultaneous requests from 20 workers resulted in an approximate rate of 171 requests per second, with zero failures, showing the scalability and stability of the lightweight graph architecture for real-time blockchain monitoring.



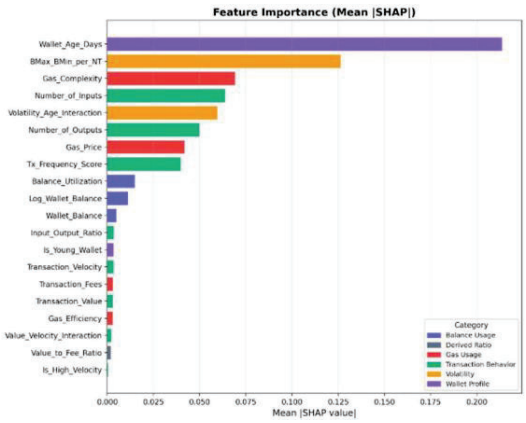
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

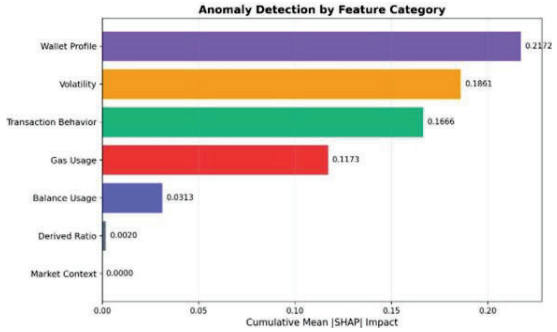




C. Explainability Results

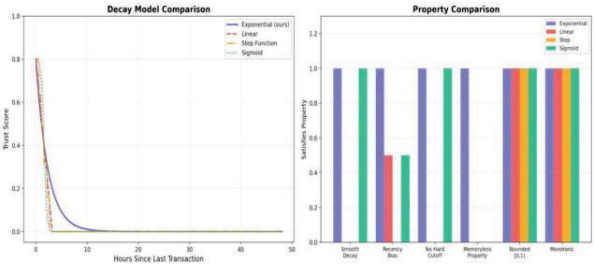
SHAP explainability results were generated for 500 transactions (randomly selected) to understand the XGBoost fraud detector. This analysis added little computational cost, taking 0.046 seconds. It found the model was largely driven by features related to wallet-profile, volatility, transaction-behaviour and gas-usage anomalies, with lesser influence from balance-utilization features. Most important individual features were Wallet_Age_Days, BMax_BMin_per_NT and Gas_Complexity. The human-readable interpretations of the SHAP values correctly identified suspicious behaviour patterns like excessive volatility, irregular transaction activity, and abnormal gas behaviour, demonstrating that the framework does not only assign a risk score, but also interpretable explanations of fraud in a meaningful way to a human analyst.

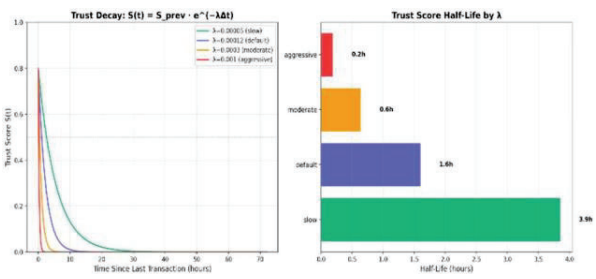




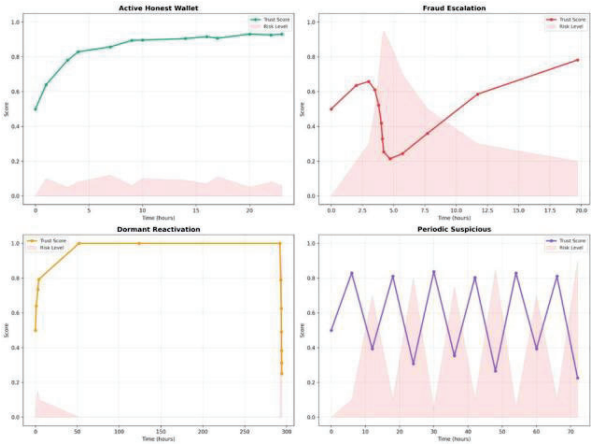
D. Temporal Trust Model

We tested the temporal trust model with an exponential decay function ($S_{decayed} = S_{prev} \times e^{(-\lambda \Delta t)}$) to model how wallet trust evolves over time under various behavioural scenarios. The default decay coefficient ($\lambda = 0.00012$) yielded a trust half-life of about 1.6 hours, enabling the system to focus on recent wallet activity, while de-emphasizing older activity. Analytical proof established desirable properties of the decay function, such as monotonicity, convexity, boundary conditions, and memorylessness. Experiments with simulated wallet lifecycles revealed different trust dynamics for wallets that were active and honest, inactive, involved in a fraud escalation, or periodically suspicious, demonstrating that the model is sensitive to changes in wallet behaviour. Benchmarking with linear, sigmoid and step-function decay models also demonstrated that exponential decay was the most reliable and realistic model for trust evolution in blockchain systems.



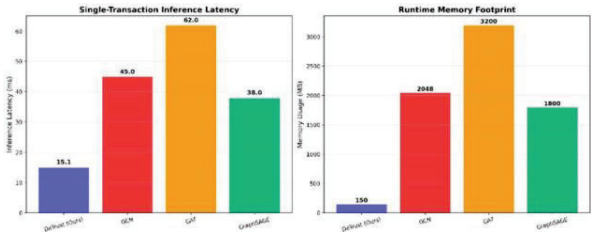


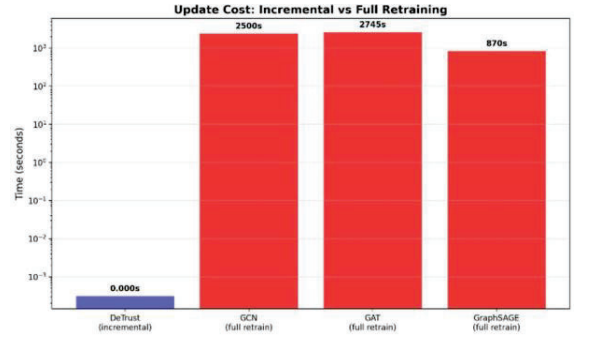
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026



E. Operational Trade-Offs

Operational evaluations benchmarked DeTrust ETH against typical GNN algorithms such as GCN, GAT and GraphSAGE using the same synthetic transaction graph. Experimental results reveal that DeTrust ETH has a median single-prediction latency of 15 ms, compared to 38-62 ms for the GNN baselines. Incremental updates to the graph take on average 0.32 ms, while GNN-based approaches require costly full-graph retraining which takes hundreds or thousands of seconds. The framework also significantly reduced memory costs (approximately 150 MB vs. 1.8-3.2 GB for GNNs) with real-time processing requirements. These results show that efficient graph analytics can offer significantly lower latency, computational costs, and incremental graph updates compared with highly computation-intensive graph-learning methods, better equip the proposed framework for real-time blockchain fraud monitoring.





Operational Comparison: Graph Analytics vs GNN

System	Inference (ms)	Memory (MB)	Retraining	Real-Time
DeTrust (Ours)	15.1	150	No	Yes
GCN	45.0	2048	Yes	No
GAT	62.0	3200	Yes	No
GraphSAGE	38.0	1800	Yes	No

VI. DISCUSSION

A. Deployment-Oriented Perspective

Our study suggests that blockchain fraud-detection systems should be evaluated based on operational features such as responsiveness, explainability, scalability and auditability, as well as predictive performance measures, such as accuracy or recall. The majority of past academic research on fraud detection has been focused on offline classification with historical blockchain transactions. However, real blockchain monitoring systems will be relying on a real time stream of transactions and the analyst needs to have a real time view of a suspicious transaction. In such settings, the speed of prediction or "black box" model outcomes can reduce practical value regardless of classification performance.

DeTrust ETH addresses these practical concerns by bringing together explainable machine learning, light-weight graph algorithms, temporal trust calculations and real time dashboard technology into a low latency platform. As the framework demonstrates, to perform real-world monitoring of blockchain fraud, a trade-off between prediction and deployability, and analyst interpretability, is required.

While Graph Neural Networks (GNNs) are a good baseline for blockchain fraud, adding them to an equivalent real-time evaluation scenario is costly. GNNs are typically based on the retraining of the whole graph and recomputation of the embedding, coupled with high GPU consumption, and are thus difficult to deploy in a streaming setting. The emphasis of our work on a deployable and fast inference system means that this work is compared with GNNs from an architectural perspective rather than equivalence. The purpose of the work is not to compete with the offline accuracy of GNNs, but to demonstrate the usefulness of lightweight graph reasoning and explainability in machine learning for fraud intelligence in streaming scenarios.

B. Related Work: GNN-Based Systems.

Recently, a lot of work has been done on using Graph Neural Networks (GNNs) to detect blockchain fraud as the blockchain can be considered a large transaction graph [5][6]. GNNs can in particular be helpful in capturing structural relationships between wallets and capturing multi-hop transactional relationships that are not captured by traditional classifiers. While such capabilities are useful, GNN-based systems have several limitations in real-time settings. Most approaches periodically retrain the model when new graph edges are added, which is a costly process. Large-scale recomputation of embeddings may also increase storage and inference time, particularly for a dynamic blockchain network. Additionally, graph embeddings may be hard to interpret, limiting their explainability to fraud analysts and compliance officers.

DeTrust ETH does not aim to completely replace GNN-based approaches. Instead, it demonstrates that simple and efficient graph traversals and risk propagation can provide valuable real-time insights into fraud, with significantly reduced costs. The proposed approach is therefore a deployment-oriented approach that can be fine-tuned to be real-time and operationally scalable rather than learning complex relational representations.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

C. Limitations

There are some limitations to the current solution. First, the system has been tested on the Ethereum Sepolia testnet instead of the Ethereum mainnet. Although the testnet provides a controlled environment in which we can carry out experiments, in the real world, a deployment will involve much higher volume of transactions, a more complex structure of the network and a wider range of fraudulent patterns.

Second, the fraud detection system was tuned to favor recall (detection of fraudulent transactions) in order to reduce the number of frauds that would be missed. This, in turn, has resulted in a relatively low precision, i.e. many of the transactions detected will be legitimate. While this may be acceptable in the context of an early warning system, humans must still be in the loop to make enforcement decisions. In addition, the current graph engine doesn't carry any learned node embeddings and global graph optimization techniques as used in more sophisticated GNNs. While this improves the system's responsiveness, the ability to learn some long-term structural dependencies may be reduced. The other components rely on external service providers such as RPC providers and blockchain APIs. Dependency on these services introduces operational risk, rate limiting and possibly service unavailability in periods of congestion.

Finally, the study is limited in the sense that the data used are publicly available and sourced from the testnet transaction feeds, and does not necessarily reflect the entire scope and complexity of fraud in Ethereum. This is because the amount of data available in blockchain systems with labels on fraud is scarce and frauds are often dynamic. This can result in imprecise labels and hence reduced precision. This will be addressed in future work by providing larger datasets from mainnet, forensic labels and human feedback to improve the precision and stability of the fraud detection model.

VII. FUTURE WORK

The current system can be further enhanced in several ways. One such future is the implementation of active learning algorithms where analysts' feedback is used to continuously train and improve the fraud-detection model. This could allow the system to adapt to new techniques and fraud patterns. Future work can also include additional research on cross-chain transaction analysis, in particular bridge-based laundering (cash-outs between chains). Since most of the current frauds involve multiple chains, a fraud intelligence system that can cover multiple chains would be desirable. As for systems, it should be able to be deployed in the Ethereum mainnet with distributed streaming and large-scale graph databases, and more complex concurrency control. The threshold optimization based on reinforcement learning can also improve the recall-precision trade-off in different operational scenarios.

One particularly exciting avenue of research is to combine periodically offline GNN-based global risk ranking with lightweight real-time graph tracing. Such designs would be able to leverage the expressivity of graph-learning algorithms and the real-time responsiveness of lightweight incremental graph analytics.

VIII. CONCLUSION

The paper presented DeTrust ETH, a production-grade fraud intelligence system on Ethereum which integrates real-time blockchain ingestion, explainable machine learning, lightweight graph analytics, temporal trust analysis, and on-chain auditability into a low-latency system. Rather than proposing a new and innovative fraud-detection technique, this work introduced a way to combine various techniques into a real-time application that can be deployed and run with continuous blockchain stream. The approach stresses responsiveness, explainability, incremental graph updates and deployability.

The results of the experiment revealed its high fraud recall, low latency graph execution, scalable parallel API execution, and explainability in real-time with SHAP-based behavioral interpretation. The lightweight graph engine was able to detect circular funding, unusual funding structures and risk propagation with no need for costly retraining of the graphs.

Unlike prior work that primarily emphasizes offline predictive accuracy, this paper introduces a **deployment-oriented fraud intelligence architecture** that is optimized for real-time blockchain environments this contribution is not limited to model selection, but lies in the coordinated integration of explainability, temporal reasoning, and graph-based risk propagation within a low-latency operational pipeline.

The findings suggest that lightweight graph intelligence together with explainable machine learning is a promising and low-cost approach to operational fraud monitoring on Ethereum. While the framework is not a replacement for the existing state-of-the-art in Graph Neural Networks, it shows that more lightweight approaches to graph reasoning can be beneficial in real-time monitoring, where interpretability, responsiveness and scalability are crucial to application.

References

[1] Aparecium: Understanding and Detecting Scam Behaviors on Ethereum via Biased Random Walk, SpringerOpen, 2023.

[2] S. Shukla and H. Sharma, Intelligent Fraud Detection in Ethereum Transactions Using Machine Learning, IJRASET, 2021.

[3] Fraud Detection in Smart Contracts Anomaly Detection, SciTePress, 2024.

[4] Lundberg, S., and Lee, S.-I., A Unified Approach to Interpreting Model Predictions, NeurIPS, 2017.

[5] Weber, M. et al., Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics, KDD Workshop, 2019.

[6] Pareja, A. et al., EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs, AAAI, 2020.

[7] A. Agarwal, A. Barve, and S. Shukla, “Detecting Malicious Accounts in Permissionless Blockchains using

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Temporal Graph Properties," *Journal of Blockchain Research and Applications*, vol. 2, no. 1, pp. 1–14, 2021.

[8] "Dynamic Trust Decay: Adaptive Profiling Mechanism for Blockchain Oracles," *ResearchGate Preprint*, 2024. [Online]. Available: https://www.researchgate.net/publication/403185657_Dynamic_Trust_Decay_Adaptive_Profiling_Mechanism_for_Blockchain_Oracles

[9] TRacer: Scalable Graph-Based Transaction Tracing for Account-Based Blockchain Trading Systems.

[10] "DenseFlow: Spotting Cryptocurrency Money Laundering in Ethereum Transaction Graphs," in *Proceedings of the Web Conference (WWW)*, 2024.

[11] Ayush Kumar and Vrizlynn L.L. Thing, "A Survey of Transaction Tracing Techniques for Blockchain Systems," *Cyber Security Strategic Technology Centre ST Engineering*.

[12] D. Bespalov, R. Brand, and Y. Qi, "Build a GNN-based real-time fraud detection solution using the Deep Graph Library without using external graph storage," Amazon Web Services (AWS), Feb. 28, 2023. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/build-a-gnn-based-real-time-fraud-detection-solution-using-the-deep-graph-library-without-using-external-graph-storage/

[13] "Cryptocurrency Scam Dataset for DQN Models," *Mendeley Data*.

[14] V. Aliyev, "Ethereum Fraud Detection Dataset," Kaggle, Jan. 2021.