ETH-TraceBench: A Large-Scale Event-Stream Benchmark for Ethereum DeFi under Temporal, Protocol, and Contract Shift

Anonymous authors
Paper under double-blind review

Abstract

Ethereum decentralized-finance (DeFi) transactions are not ordinary table rows: their economic meaning often depends on nested calls, emitted event topics, token-flow structure, call depth, gas use, and block-local order. Existing blockchain-learning benchmarks can therefore reward memorization of contract addresses, protocol tags, or generic event signatures rather than transferable trace understanding. We introduce ETH-TraceBench, a benchmark for trace-native representation learning on Ethereum DeFi event streams under naturally occurring temporal, protocol, contract, and topic shift. The raw event universe covers Ethereum mainnet from January 2021 through December 2025 and contains 1.35 billion transactions with logs and 5.01 billion raw log rows. The task-labelled benchmark contains 311.97 million labelled transaction instances and 2.33 billion matched raw logs across a primary DEX swap/trade classification task and a liquidation rare-event stress test. The forward split trains on 2021–2024, validates on 2025H1, and tests on 2025H2. The test set includes meaningful transfer settings: Uniswap v4 and Ekubo v1 are absent from training but appear in validation and test, and 26.2 million DEX test transactions occur on pool or liquidity-infrastructure identifiers absent from training. We report verified shallow trace-count, linear topic/emitter, and lightweight neural topic/emitter baselines regenerated from transaction-level prediction artifacts. On fixed supervised evaluation samples, the strongest trace-count baseline reaches 0.953 macro-F1 for DEX classification and 0.845 macro-F1 for liquidation detection; TopicEmitterTrace-SGD reaches 0.924 macro-F1 for DEX and 0.943 macro-F1 for liquidation; and TopicEmitterHashMLP reaches 0.959 macro-F1 for DEX and 0.938 macro-F1 for liquidation. An OOD slice audit shows that symbolic baselines degrade on transactions containing unseen topics or unseen topic–emitter pairs. These results establish ETH-TraceBench as a shortcut-aware benchmark in which aggregate performance must be interpreted alongside temporal, holdout, masking, and symbolic-novelty evaluations. We specify Masked Event Modeling for Blockchain Logs (MEM-BL) as a reference protocol rather than as a reported model-superiority claim.

Abbreviations

DeFi: decentralized finance; EVM: Ethereum Virtual Machine; MEM-BL: Masked Event Modeling for Blockchain Logs; AMM: automated market maker; ABI: application binary interface.

1 Introduction

Ethereum decentralized finance (DeFi) produces a large public record of executable economic activity. Each transaction is time-stamped, globally ordered, and accompanied by emitted log events. This transparency makes Ethereum a natural setting for machine learning on financial infrastructure. It also makes the setting difficult. The economic meaning of a transaction is rarely contained in a single decoded row. A transaction can combine a swap, flash loan, collateral seizure, liquidation sale, oracle read, arbitrage leg, fee extraction,

and repayment within one atomic execution. Its interpretation depends on the sequence of emitted topics, emitting contracts, token transfers, nested calls, protocol versions, and neighboring transactions in the same block.

Execution-level composition creates a benchmark problem that is distinct from standard transaction classification or graph learning. Existing blockchain-analytics pipelines often reconstruct DeFi behavior with protocol-specific parsers, application binary interfaces, contract registries, source-code inspection, balance reconciliation, and hand-coded templates (Kalodner et al., 2017; Zhang et al., 2020; Wu et al., 2024; Chen et al., 2024). Such tools are useful for forensic interpretation, but they are fragile as learning benchmarks. They can fail when protocols upgrade through proxies, forks clone existing contracts, aggregators route across venues, or new DeFi mechanisms appear before stable labels exist (Schar, 2021; Werner et al., 2022; Qin et al., 2021; Zhou et al., 2021). At the same time, a naive machine-learning benchmark can be too easy: models may memorize event signatures, pool addresses, protocol tags, or generic token-transfer topics rather than learn transferable event-stream structure (Weber et al., 2019; Bellei et al., 2024; Cheng et al., 2024; Duan et al., 2026).

We introduce ETH-TraceBench, a large-scale benchmark for Ethereum DeFi event streams. The benchmark represents each transaction as a raw-log trace and evaluates models under temporal, protocol, contract, and shortcut-distribution shift. The raw event universe covers Ethereum mainnet from January 2021 through December 2025 and contains 1.35 billion transactions with logs and 5.01 billion raw log rows. The task-labelled benchmark contains 311.97 million labelled transaction instances and 2.33 billion matched raw logs. It includes a primary DEX swap/trade classification task and a liquidation rare-event stress test.

The main evaluation split is chronological. Training uses 2021–2024, validation uses 2025H1, and testing uses 2025H2. The chronological design reduces random-split leakage and creates a forward deployment setting. The test period differs from training in measurable ways: DEX test transactions contain more raw logs, more unique event topics, and more emitting contracts per transaction than the training period. Protocol-transfer and contract-holdout conditions are also included. Uniswap v4 and Ekubo v1 are absent from the 2021–2024 training period but appear in 2025 validation and test. At the pool or liquidity-infrastructure level, 26.2 million DEX test transactions occur on identifiers absent from training.

ETH-TraceBench is motivated by the evaluation concern that underlies recent benchmark work on real-world distribution shift: strong performance on random or identifier-overlapping splits may not imply deployable generalization. In WILDS, each example is associated with a domain and evaluation focuses on naturally occurring domain and subpopulation shifts rather than only independent and identically distributed splits (Koh et al., 2021). In EarthquakeNPP, the benchmark contribution is not only a dataset but also a correction to leakage-prone splits, missing extreme sequences, and inadequate reference baselines in prior earthquake forecasting evaluations (Stockman et al., 2026). ETH-TraceBench applies the same benchmark principle to Ethereum DeFi by treating protocol versions, contract identities, event topics, time periods, and shortcut views as part of the evaluation design rather than as incidental metadata.

We report three verified baseline layers. First, aggregate trace-count baselines use only raw-log count, unique-topic count, and unique-emitter count. High predictive power from these simple features shows that execution-complexity shortcuts must be measured rather than ignored. Second, Ethereum-native topic/emitter baselines use event-topic bags, emitter-address bags, and topic–emitter combinations. The symbolic baselines test whether event signatures alone are sufficient for generalization. Third, a lightweight neural topic/emitter baseline embeds hashed symbolic event features and provides a reproducible neural reference point. Slice-level audits further compare performance on examples whose topics or topic–emitter pairs were seen during training with examples containing unseen symbolic elements. The audits show that high aggregate topic/emitter performance can mask shortcut-sensitive degradation.

The paper makes four contributions. First, it defines Ethereum DeFi learning as transaction-level event-stream learning under temporal, protocol, contract, and shortcut-distribution shift. Second, it contributes a verified benchmark construction protocol with chronological splits, protocol-transfer manifests, pool/liquidity-infrastructure holdouts, raw-log coverage diagnostics, and generic-topic exclusions. Third, it reports verified baseline results from transaction-level prediction artifacts, including trace-count baselines, topic/emitter baselines, a lightweight neural topic/emitter baseline, and OOD slice audits. Fourth, it specifies

a reproducibility and leaderboard protocol requiring temporal, protocol-transfer, contract-holdout, address-masked, topic-masked, and address+topic-masked reporting. No numerical performance claim is reported unless it can be regenerated from included prediction artifacts. MEM-BL is retained as a reference modeling protocol rather than presented as a model-superiority result.

Figure 1 summarizes the benchmark pipeline. The central contribution is an auditable construction and evaluation protocol for trace-native DeFi learning rather than a single classifier for one protocol-specific task.

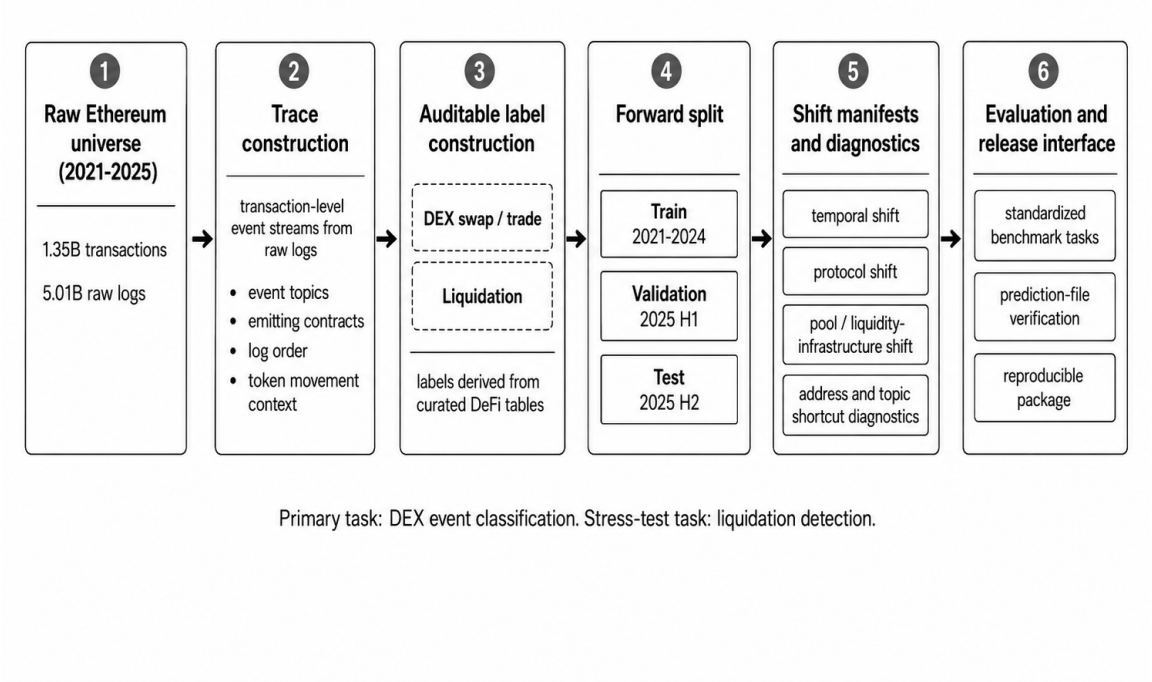

Figure 1: ETH-TraceBench benchmark construction and evaluation pipeline. The pipeline begins from the raw Ethereum log universe, constructs transaction-level event streams, derives auditable DEX and liquidation labels, imposes a forward train/validation/test split, generates shift and shortcut diagnostics, and specifies a reproducible evaluation and release interface.

2 Related Work and Benchmark Positioning

2.1 Benchmark papers for real-world shift

ETH-TraceBench is positioned as a benchmark paper rather than as a blockchain application note or a model-superiority paper. WILDS argues that real-world distribution shifts are underrepresented in widely used machine-learning datasets and designs benchmark datasets around naturally occurring domain and subpopulation shifts (Koh et al., 2021). The relevant lesson for Ethereum is that a chronological split is not enough: the benchmark must document what changes between training and test and whether the shift is meaningful for deployment. EarthquakeNPP provides a second model. It identifies flaws in existing neural point-process earthquake benchmarks, including leakage-prone splits and omission of a major sequence, and introduces a curated benchmark platform with domain-relevant baselines and evaluation protocols (Stockman et al., 2026). The relevant lesson for this paper is that benchmark value does not require a new neural model to outperform every baseline; the benchmark can be valuable because it standardizes the problem, documents leakage and shortcut risks, and creates a credible evaluation platform.

Prior blockchain machine-learning and analytics work has advanced address-graph forensics, smart-contract vulnerability detection, transaction misbehavior analysis, MEV measurement, and DeFi attack detection (Weber et al., 2019; Bellei et al., 2024; Elmougy et al., 2023; Cheng et al., 2024; Duan et al., 2026; Ashizawa

et al., 2022; Wang et al., 2025; Daian et al., 2020; Qin et al., 2022; Wu et al., 2024). Existing papers rarely provide this combination of scale, forward splits, protocol-transfer manifests, contract-holdout structure, and shortcut diagnostics. ETH-TraceBench therefore complements generic distribution-shift benchmarks with a public, high-frequency, executable financial infrastructure setting whose events are typed, compositional, and strategically generated.

The benchmark gap is not merely the absence of a larger Ethereum dataset. It is a set of recurring evaluation failures visible across transaction-table, graph-forensics, smart-contract, and DeFi-security settings. Transaction-table benchmarks often collapse a multi-log execution into one row and therefore remove the event-stream structure that a trace model should learn (Meiklejohn et al., 2013; Ron et al., 2013; Kalodner et al., 2017; Weber et al., 2019). Random splits often place the same contracts, pools, token contracts, and event topics in both train and test, which can turn deployment generalization into identifier replay (Koh et al., 2021; Stockman et al., 2026). Protocol-specific decoded tables provide useful labels, but if they are used as model inputs they can leak the answer through parser output (Zhang et al., 2020; Wu et al., 2024; Chen et al., 2024). Finally, many blockchain classifiers report aggregate test performance without separating temporal shift, new-protocol transfer, pool holdout, and shortcut-controlled views (Weber et al., 2019; Bellei et al., 2024; Elmougy et al., 2023; Cheng et al., 2024; Duan et al., 2026). ETH-TraceBench is designed to make these failure modes measurable rather than implicit.

2.2 Audit of existing blockchain benchmark families

A useful benchmark paper should make the missing benchmark object explicit. EarthquakeNPP does this by identifying leakage-prone splits, omitted high-impact sequences, and weak baseline comparisons in prior NPP earthquake benchmarks; WILDS does this by showing that standard i.i.d. benchmarks underrepresent naturally occurring deployment shifts. We follow the same logic by auditing the benchmark families commonly used in blockchain machine learning and DeFi analytics. Table 1 summarizes why these resources are valuable, what they miss for Ethereum DeFi monitoring, and how ETH-TraceBench fills the gap. Table 2 summarizes the benchmark card used by the rest of the paper. Table 3 summarizes the formal shift settings and the verified evidence attached to each setting. Table 5 illustrates why event-name or parser shortcuts are insufficient for trace-native interpretation. Table 7 reports the raw-log universe used to construct the benchmark. Table 8 reports the task-labelled benchmark scale. Table 9 reports the final split summary and raw-log coverage diagnostics by task. Table 10 summarizes the DEX pool and liquidity-infrastructure holdout. Table 11 reports event-topic shortcut diagnostics for the 2025H2 test split. Table 14 lists the recommended leaderboard tasks. Table 16 defines the Ethereum-native topic/emitter baseline protocol. Table 19 reports forward-shift diagnostics before model evaluation. Table 20 reports benchmark-level shortcut diagnostics. Table 22 reports the verified Ethereum-native topic/emitter baseline results. Table 23 reports the verified lightweight neural topic/emitter baseline.

2.3 Ethereum execution traces and DeFi analytics

Prior research shows that Ethereum and DeFi monitoring now require methods that move across economics, security, privacy, and execution analysis. Hickey and Harrigan study the Bisq decentralized exchange and show that participation in decentralized markets can carry measurable privacy costs (Hickey et al., 2022). Laurent et al. analyze Ethereum transaction-fee optimization, foregrounding the operational role of gas prices and transaction timing (Laurent et al., 2022). Ashizawa et al. introduce Eth2Vec for contract-wide code representation learning in Ethereum vulnerability detection, while Chen et al. map blockchain-service security from a full-stack architecture perspective (Ashizawa et al., 2022; Chen et al., 2023). Buterin et al. develop Privacy Pools as a practical equilibrium between blockchain privacy and regulatory compliance (Buterin et al., 2024). Huang et al. propose TMAS for transaction misbehavior analysis using graph features, money-laundering heuristics, and account linkage, and Gan et al. develop FinanceFuzz to test smart contracts against financial properties (Huang et al., 2024; Gan et al., 2025). Subsequent research extends Ethereum analysis to sound EVM bytecode control-flow construction, graph-and-temporal smart-contract vulnerability detection, and MEV formalization (Arceri et al., 2025; Wang et al., 2025; Alnajjar et al., 2026).

Table 1: Audit of existing blockchain benchmark families and the ETH-TraceBench gap

Benchmark family	What it captures well	What it misses for DeFi event-stream learning	ETH-TraceBench response
Address or transaction graph benchmarks	Ledger-scale address interaction, illicit-activity labels, graph structure, and temporal flows.	Transaction-internal execution semantics: emitted topics, call depth, multi-contract composition, and token-flow direction inside one atomic transaction.	Uses raw logs as typed event streams and evaluates graph/address information as a baseline rather than as the whole benchmark object.
Decoded protocol tables	High-precision application activity such as swaps, trades, loans, or liquidations.	They are labels and metadata, not the input representation; using them directly can reward protocol-tag memorization.	Uses curated tables to create auditable labels, while inputs are raw logs and diagnostics quantify protocol and topic shortcuts.
Random-split transaction classifiers	Convenient supervised learning setup and large sample sizes.	Inflated estimates when training and test share the same time periods, protocols, pool contracts, and event-topic distribution.	Enforces a forward split: train on 2021–2024, validate on 2025H1, and test on 2025H2.
Smart-contract code or bytecode benchmarks	Program-level vulnerability and code-family patterns.	Runtime economic behavior that emerges from interactions among contracts, users, pools, routers, and liquidators.	Treats each transaction as an executed trace and combines event syntax, token-flow structure, and block-local ordering.
Attack or anomaly datasets	Specific exploit, scam, laundering, or misbehavior labels.	Narrow event types; limited coverage of routine DeFi activity; often not designed for protocol or contract holdout.	Provides a broad DEX event task plus liquidation rare-event stress test, with protocol-transfer and pool/liquidity-infrastructure holdout manifests.
Parser- or rule-based DeFi studies	Interpretability and high precision for known mechanisms.	Brittleness under forks, proxy upgrades, aggregator routing, and new mechanisms without stable decoders.	Frames parsers as label sources and baselines, while requiring models to report shortcut-controlled evaluation under address/topic masking.

The paper builds on that research trajectory but targets a different data layer. Existing studies primarily examine exchange-level behavior, fee optimization, privacy protocols, account-level transaction misbehavior, bytecode-level program analysis, vulnerability detection, or formal MEV definitions. MEM-BL instead learns from transaction-internal execution traces: receipts, emitted logs, internal calls, token transfers, gas information, call depth, and block-local ordering. The model therefore treats each Ethereum transaction as a nested, typed, multi-contract event stream whose economic meaning may only become visible after event syntax, token-flow direction, call-depth structure, and neighboring transaction context are combined.

Broader DeFi research has documented automated market makers, flash loans, liquidations, MEV, transaction reordering, and composable financial protocols (Schar, 2021; Werner et al., 2022; Daian et al., 2020; Qin et al., 2021; 2022; Zhou et al., n.d.; 2021; Lehar et al., 2025). Automated market makers and liquidity protocols introduce event patterns whose economic meaning depends on pool state, token direction, reserve changes, and transaction timing (Angeris et al., 2020; 2021; Milionis et al., 2022; Park, 2023). Liquidation and MEV behavior often depends on ordering and flow structure rather than isolated event names (Daian et al., 2020; Qin et al., 2022; Weintraub et al., 2022; Alipanahloo et al., 2024; Materwala et al., 2024).

Trace-based security systems show why low-level execution data matter. TXSPECTOR uncovers Ethereum attacks from transaction traces, while DeFiRanger detects DeFi price manipulation attacks (Zhang et al., 2020; Wu et al., 2024). Related smart-contract analysis methods learn from bytecode, transaction behavior, or invariants (Ashizawa et al., 2022; Chen et al., 2024). MEM-BL differs from these systems because it learns reusable transaction representations before a specific attack, protocol, or downstream label is fixed.

2.4 Blockchain transaction graphs and financial forensics

Public ledger data have long supported empirical blockchain measurement and financial forensics. Early Bitcoin studies transformed public ledgers into transaction graphs and address clusters (Meiklejohn et al., 2013; Ron et al., 2013). BlockSci provided a scalable platform for blockchain analysis (Kalodner et al., 2017). Anti-money-laundering benchmarks and graph-learning studies such as Elliptic, Elliptic++, and Elliptic2 demonstrate the value of public transaction data for illicit-activity detection and financial forensics (Weber et al., 2019; Bellei et al., 2024; Elmougy et al., 2023). Recent cryptocurrency data-mining studies detect malice, illicit addresses, and financial fraud in dynamic graph streams (Cheng et al., 2024; Duan et al., 2026; Khodabandehlou et al., 2024).

MEM-BL shifts the unit of analysis from address-level Bitcoin graphs or trader-level dynamic graphs to Ethereum transaction-internal execution traces. Address and transfer graphs remain useful baselines, but DeFi mechanisms often unfold within one transaction through nested calls and emitted events. A graph can show that assets moved between addresses. The graph may not distinguish a user swap, flash-loan-assisted liquidation, arbitrage leg, aggregator route, or just-in-time liquidity strategy unless call depth, event order, token-flow direction, and block-local context are modeled jointly.

2.5 Self-supervised learning for logs and structured event streams

Self-supervised learning motivates MEM-BL's pretraining design. Transformer models learn long-range dependencies in sequences (Vaswani et al., 2017). BERT introduced masked language modeling for bidirectional representation learning (Devlin et al., 2019), and masked autoencoders extended reconstruction-based pretraining to visual patches (He et al., 2022). System-log models provide a closer technical analogy. DeepLog models machine logs as sequences for anomaly detection, and LogBERT adapts masked pretraining to log anomaly detection (Du et al., 2017; Guo et al., 2021). Self-supervised representation learning for sparse and irregular clinical time series also shows how pretraining can improve irregular structured data before task labels are introduced (Tipirneni et al., 2022).

Blockchain traces differ from ordinary logs in four ways. First, their syntax is executable because events arise from smart-contract execution. Second, their semantics are economic because emitted logs and transfers express swaps, liquidations, loans, collateral movements, and pool-state changes. Third, their producers are strategic because users, searchers, builders, contracts, and protocols act under incentives created by ordering and composability. Fourth, their identifiers are both informative and dangerous because contract addresses, event signatures, and protocol tags can create shortcuts that overstate generalization. MEM-BL therefore adapts self-supervised learning to event topics, typed attributes, call depth, token-flow consistency, and block-local ordering.

2.6 Dynamic graph and temporal representation learning

Graph and temporal representation learning provide important baselines for blockchain analytics. GraphSAGE introduced inductive graph representation learning, Graph Attention Networks use attention over graph neighborhoods, Temporal Graph Networks maintain dynamic node memories, and TGAT uses temporal attention for inductive representation learning on temporal graphs (Hamilton et al., 2017; Velickovic et al., 2018; Rossi et al., 2020; Xu et al., 2020). Fraud-oriented graph mining also shows that financial graph models must be robust against camouflage (Hooi et al., 2017). These methods motivate address-contract-transfer graph baselines and contract-holdout evaluation. MEM-BL complements this literature by modeling the event sequence inside each transaction and the local ordering around the focal transaction. The framework therefore uses graph-inspired context but does not reduce the blockchain data object to edges alone.

3 ETH-TraceBench Design Criteria and Problem Definition

Following benchmark-design principles from WILDS and EarthquakeNPP, the benchmark is built around five criteria. First, the split should capture naturally occurring distribution shift rather than a synthetic perturbation or random partition. Second, the task should be relevant to deployed monitoring systems, where future protocols, contracts, and event vocabularies differ from those observed during training. Third, the shift should be nontrivial but not impossible: training data should contain enough repeated execution structure to support transfer. Fourth, the benchmark should diagnose shortcuts from addresses, protocol tags, and event topics rather than assuming that high performance reflects reusable structure. Fifth, the package should be auditable: labels, split manifests, holdout identifiers, shortcut diagnostics, and model predictions should be regenerable from documented artifacts.

3.1 Nested strategic event streams

Let a blockchain dataset consist of blocks indexed by b. Each block contains an ordered sequence of transactions, and each transaction contains one or more external and internal calls. Each call may emit one

Table 2: ETH-TraceBench benchmark card

Item	ETH-TraceBench specification
Domain	Ethereum decentralized finance event streams
Raw universe	1.35B Ethereum transactions with logs; 5.01B raw log rows; January 2021–December 2025
Labelled benchmark	311.97M labelled transaction instances; 2.33B matched raw logs
Primary task	DEX swap/trade event classification
Stress-test task	Lending-liquidation detection
Train/validation/test	2021–2024 / 2025H1 / 2025H2
Shift axes	Temporal shift, protocol transfer, pool/liquidity-infrastructure holdout, address masking, topic masking, generic-topic exclusion
Known shortcut risks	ERC-20 Transfer, Approval, common Swap/Sync topics, common token contracts, protocol and pool identifiers
Required model artifacts	Transaction-level predictions with task, split, condition, model, seed, true label, predicted label, score, and holdout metadata

or more logs. We write $D = \{B_b\}$, where $B_b = (T_{b1}, \ldots, T_{bm})$. Each transaction is a sequence of events $T_{bj} = (e_{bj1}, \ldots, e_{bjn})$.

Each event contains typed attributes

$$e_i = (s_i, c_i, u_i, v_i, a_i, x_i, r_i, d_i, p_i, \tau_i, g_i), \tag{1}$$

where s_i is the event signature or raw topic, c_i is the emitting contract, u_i and v_i are sender, receiver, caller, or callee addresses when available, a_i is the token or asset identifier, x_i is a numerical amount, r_i is a weak role tag, d_i is call depth, p_i records within-transaction and within-block position, τ_i records timestamp or block gap, and g_i records gas-related information.

The central monitoring problem is to learn a transaction representation z_j that captures event syntax, token-flow structure, call-depth context, and block-local ordering while remaining useful under protocol transfer, contract holdout, identifier masking, and partial label availability.

3.2 Problem settings and shift definitions

The benchmark follows the distribution-shift framing used in real-world benchmark suites, but instantiates the domain variable with blockchain-native metadata. Each supervised instance is written as

$$\xi_i = (x_i, y_i, t_i, d_i, v_i), \tag{2}$$

where x_i is the raw transaction trace, y_i is the weak label for a monitored DeFi event family, t_i is block time, $d_i = (d_i^{protocol}, d_i^{pool}, d_i^{contract}, d_i^{topic})$ records protocol, liquidity-infrastructure, contract, and event-topic metadata, and v_i records the evaluation view made available to the model. The view variable distinguishes full traces from address-masked, topic-masked, address+topic-masked, and generic-topic-excluded traces. The benchmark distribution can be written as a mixture over blockchain-native domains,

$$P^{split}(x, y, d, v) = \sum_{d \in \mathcal{D}} q_d^{split} P_d(x, y, v), \tag{3}$$

where train, validation, and test splits differ in mixture weights and, for holdout tests, in domain support. This notation clarifies the benchmark target. A model should not only maximize average performance on the full test split; it should also maintain performance when the test distribution places mass on future periods, protocols absent from training, pool identifiers absent from training, or views in which address and topic shortcuts have been masked.

Table 4 makes the WILDS-style evaluation logic explicit. The full 2025H2 temporal test is the verified first OOD setting in this version. Protocol-transfer, pool-holdout, and masking gaps are defined as required reporting objects; this submission reports the full temporal split and the symbolic seen/unseen slice audit, while leaving the remaining restricted prediction artifacts as explicitly specified leaderboard outputs.

Table 3: Formal shift settings in ETH-TraceBench

Setting	Domain definition	Evaluation rule	Verified benchmark evidence
Forward temporal shift	d^{time} separates 2021–2024, 2025H1, and 2025H2	Train on 2021–2024, select on 2025H1, report final scores on 2025H2	DEX test traces have higher average logs, topics, and emitters per transaction than training traces
Protocol transfer	$d^{protocol}$ is the project/protocol combination	Evaluate separately on protocol/project combinations absent from the training period	Uniswap v4 and Ekubo v1 are absent from training and account for large 2025H2 test mass
Pool or liquidity-infrastructure holdout	d^{pool} is the pool contract or liquidity-infrastructure identifier	Evaluate on test transactions whose identifier has zero training-period observations	26.2M DEX test transactions use pool or liquidity-infrastructure identifiers absent from training
Shortcut-controlled views	d^{view} records the feature view made available to the model	Re-evaluate after masking addresses, event topics, or both	Generic Transfer, Approval, Swap, and Sync topics explain large raw concentration but DEX residual concentration falls after exclusion
Rare-event stress test	The label family changes from high-volume DEX activity to lending liquidations	Report AUPRC, rare-class recall, and shortcut-controlled variants separately	Liquidations have far fewer labelled transactions and stronger residual topic concentration than DEX

The formalization separates three concepts that are sometimes conflated in blockchain evaluation. A *label family* defines what the model predicts, such as DEX swap/trade activity or liquidation activity. A *domain* defines the source of distribution shift, such as time period, protocol family, or pool identifier. A *view* defines what information is exposed to the model, such as raw identifiers, address-masked traces, topic-masked traces, or address+topic-masked traces. Reporting these separately prevents a benchmark from appearing stronger merely because a model learns stable public identifiers.

3.3 Downstream monitoring tasks

The verified benchmark contains two primary labelled tasks and several transfer diagnostics. The first task is DEX event classification. It identifies transactions that participate in decentralized-exchange swap or trade activity using high-precision labels derived from curated DEX trade tables. This task is the main large-scale representation-learning benchmark because it contains hundreds of millions of labelled transactions, billions of matched raw logs, substantial temporal shift, new-protocol exposure, and large pool or liquidity-infrastructure holdouts.

The second task is liquidation detection. It identifies transactions associated with lending-liquidation events using high-precision labels derived from curated liquidation tables. This task is much smaller and more concentrated than DEX classification, but it is valuable as a rare-event stress test. Liquidation transactions have many logs per transaction, multiple emitting contracts, and economically meaningful token-flow structure. At the same time, topic concentration is higher than in the DEX task, so results on this task must be interpreted together with address-masked, topic-masked, and generic-topic-excluded diagnostics.

The remaining evaluations are not separate label families. They are stress tests applied to the labelled tasks: temporal generalization from 2021–2024 to 2025, protocol transfer to protocols absent from training, pool or liquidity-infrastructure holdout, identifier-masked evaluation, topic-masked evaluation, and low-label

Table 4: ID/OOD and shortcut-gap reporting plan

Gap type	Reference condition	OOD or controlled condition	Artifact status
Forward temporal gap	2025H1 validation after training on 2021–2024	2025H2 test	Verified for trace-feature, linear topic/emitter, and lightweight neural baselines in Table 24
Protocol-transfer gap	Full 2025H2 test	2025H2 transactions from protocols absent in training	Defined by verified protocol manifests; reported from the same prediction files on the protocol-transfer subset
Pool-holdout gap	Full 2025H2 DEX test	DEX test transactions on pool or liquidity-infrastructure identifiers absent in training	Defined by verified holdout manifests; reported from the same prediction files on the pool-holdout subset
Address-shortcut gap	Full trace view	Contract/address-masked trace view	Required reporting axis; prediction artifact not included in this submission
Topic-shortcut gap	Full trace view	Topic-masked or generic-topic-excluded view	Required reporting axis; generic-topic concentration diagnostics verified
Combined shortcut gap	Full trace view	Address+topic-masked view	Required before claiming transferable trace-semantic representation learning

Table 5: Why event-level shortcuts are insufficient

Observable trace pattern	Why a parser or event-name shortcut can fail	Structure needed for disambiguation
Transfer → Swap → Transfer	Can be a user swap, an arbitrage leg, a liquidation sale, or an aggregator route.	Call tree, token-flow direction, amount scale, and surrounding events.
Mint → Swap → Burn	Can be routine liquidity management or just-in-time liquidity around a trade.	Block-local order, victim adjacency, and pool-state changes.
Borrow or repay events with collateral transfers	Can be ordinary repayment, refinancing, flash-loan-assisted liquidation, or cascade-linked liquidation.	Repayment source, collateral movement, call depth, and AMM sale path.
Large swap near arbitrage-like trades	Can be benign arbitrage or an observable MEV pattern.	Transaction adjacency, price-impact evidence, and multi-transaction flow.
Proxy contract emits non-standard topics	Decoded name may be unavailable or misleading after upgrade.	Bytecode family, implementation history, raw topics, and flow reconciliation.

evaluation. This organization keeps the supervised problem aligned with auditable labels while still testing the generalization properties that matter for TMLR-style representation-learning evaluation.

4 Reference Modeling Protocol: Masked Event Modeling for Blockchain Logs

MEM-BL is included as a reference modeling protocol for leaderboard-style evaluation on ETH-TraceBench. MEM-BL is not presented as an empirically superior model because the package does not contain saved prediction artifacts. The purpose of the section is to define a trace-native encoder and pretraining objective that future experiments can implement and compare against parser, feature, sequence, graph, and generic Transformer baselines under the same temporal, holdout, and masking conditions.

4.1 Event representation and encoder

Each transaction is represented as a block-local execution window rather than as an isolated row in a transaction table. For event e_i in transaction T_j, MEM-BL forms a structured embedding by summing embeddings for event topic, emitting contract, asset, role, position, call depth, elapsed block time, amount bin, gas features, and a vector of normalized continuous covariates:

$$x_i = E_s(s_i) + E_c(c_i) + E_a(a_i) + E_r(r_i) + E_p(p_i) + E_d(d_i) + E_\Delta(\Delta_i) + E_x(b(x_i)) + E_g(g_i) + W_q q_i. \tag{4}$$

Here s_i is a decoded event signature or raw topic hash, c_i is the emitting contract or proxy-resolved implementation family, a_i is the token or asset identifier, r_i is a weak economic role tag, p_i records within-transaction and within-block position, d_i is call depth, $b(x_i)$ is a logarithmic amount bin, and q_i contains continuous features such as token entropy, contract age, address activity, counterpart diversity, and gas ratios.

The encoder has three levels. A transaction-local Transformer maps event embeddings into contextual event states

$$h_{ij} = f_\theta^{\text{event}}(x_{1:n_j}). \tag{5}$$

Attention pooling produces a focal transaction representation

$$z_j = \sum_{i=1}^{n_j} \alpha_{ij} h_{ij}, \qquad \alpha_{ij} = \frac{\exp(u^\top \tanh(W h_{ij}))}{\sum_{k=1}^{n_j} \exp(u^\top \tanh(W h_{kj}))}. \tag{6}$$

A block-context Transformer contextualizes z_j with neighboring transactions in the same block. A lightweight address-contract memory stores time-stamped summaries of recent contract and address behavior. The memory is queried only with information observable at or before the focal block, which prevents future-information leakage.

4.2 Blockchain-specific pretraining objectives

Pretraining corrupts a transaction window W_j and asks the encoder to reconstruct missing execution structure. The total objective is

$$\mathcal{L} = \mathcal{L}_{\text{topic}} + \lambda_a \mathcal{L}_{\text{attr}} + \lambda_c \mathcal{L}_{\text{ctx}} + \lambda_o \mathcal{L}_{\text{ord}} + \lambda_f \mathcal{L}_{\text{flow}}. \tag{7}$$

Masked topic prediction reconstructs masked decoded signatures or raw topic hashes:

$$\mathcal{L}_{\text{topic}} = -\sum_{i \in M_s} \log p_\theta(s_i \mid \widetilde{W}_j). \tag{8}$$

Masked attribute prediction reconstructs masked contracts, assets, role tags, and amount bins:

$$\mathcal{L}_{\text{attr}} = -\sum_{i \in M_a} \sum_{q \in \{c,a,r,b(x)\}} \log p_\theta(q_i \mid \widetilde{W}_j). \tag{9}$$

Transaction-context reconstruction removes an event span S_i and identifies the true span from a candidate set $\mathcal{C}_i$ that includes hard negatives from similar protocols, forks, or bytecode families:

$$\mathcal{L}_{\text{ctx}} = -\sum_{i \in M_c} \log \frac{\exp(\text{sim}(z_j, z_{S_i})/\tau)}{\sum_{S' \in \mathcal{C}_i} \exp(\text{sim}(z_j, z_{S'})/\tau)}. \tag{10}$$

Order-consistency prediction detects whether event spans or neighboring transactions have been permuted:

$$\mathcal{L}_{\text{ord}} = -\sum_{i \in M_o} \left[y_i^o \log \hat{y}_i^o + (1 - y_i^o) \log(1 - \hat{y}_i^o) \right]. \tag{11}$$

Flow-consistency prediction detects token-flow corruptions produced by replacing an asset identifier, changing an amount bin, or reversing a transfer direction:

$$\mathcal{L}_{\text{flow}} = -\sum_{i \in M_f} \left[y_i^f \log \hat{y}_i^f + (1 - y_i^f) \log(1 - \hat{y}_i^f) \right]. \tag{12}$$

Protocol-aware masking reduces shortcut learning by masking contract identifiers, decoded event names, and protocol tags together when they would otherwise reveal the downstream class. Hard negatives force the model to distinguish economically different traces that share superficial event names.

4.3 Fine-tuning and inference

For each downstream task, a task head g_ϕ maps the transaction representation to a label distribution:

$$p(y_j \mid T_j) = g_\phi(z_j). \tag{13}$$

Fine-tuning minimizes cross-entropy on the training split and selects checkpoints using validation macro-F1:

$$\mathcal{L}_{\text{task}} = -\sum_{j \in D_{\text{train}}} \sum_{k=1}^{K} \mathbf{1}\{y_j = k\} \log p_\phi(y_j = k \mid T_j). \tag{14}$$

We evaluate frozen, partially fine-tuned, and fully fine-tuned variants. Frozen encoders test whether pretraining alone produces transferable representations. Partial fine-tuning updates the block-context and task-head layers. Full fine-tuning updates all parameters. Scores are calibrated with validation-set temperature scaling when calibration affects the reported metric.

5 Data and Benchmark Construction

A benchmark for blockchain trace representation learning must do more than collect large numbers of transactions. It must specify the raw event universe, document how labels are produced, prevent future leakage, expose naturally occurring shift, and quantify shortcuts created by addresses and event topics. Table 6 maps the benchmark-design criteria to the diagnostics implemented in ETH-TraceBench.

5.1 Ethereum raw-log universe

The empirical setting is Ethereum mainnet from January 2021 through December 2025. The data construction uses public chain-derived tables for raw logs, decentralized-exchange trades, and lending liquidations. The benchmark is deliberately organized in layers. The raw-log universe defines the self-supervised representation-learning environment. The weak-label sources define auditable downstream labels. The temporal, protocol-transfer, contract-holdout, and shortcut-diagnostic splits define the evaluation setting.

The raw-log audit covers 1,346,806,238 transactions that emit at least one log and 5,013,719,926 raw log rows. The distribution is highly skewed. Most transactions emit only one or two logs, but most log mass appears in richer multi-event traces. Transactions with three or more logs account for 80.6% of all raw log rows, and the small tail of transactions with 51 or more logs accounts for 738.7 million raw log rows.

Table 6: Benchmark design requirements and implemented diagnostics

Requirement	Implementation in the benchmark
Scale and raw-event coverage	Raw-log audit over 1.35B Ethereum transactions and 5.01B raw log rows from 2021–2025.
Auditable labels	High-precision labels from curated DEX trade and lending-liquidation tables; labels are treated as task labels rather than a full taxonomy of Ethereum behavior.
Forward generalization	Train on 2021–2024, validate on 2025H1, test on 2025H2.
Protocol and contract shift	New-protocol and pool/liquidity-infrastructure holdout manifests quantify exposure to identifiers absent from training.
Shortcut controls	Contract-concentration, topic-concentration, common-token exclusion, and generic-topic exclusion diagnostics.
Reproducibility	Monthly query chunks, split manifests, label manifests, and raw-log join diagnostics.

Table 7: Raw Ethereum log universe, 2021–2025

Event-complexity bin	Transactions	Raw log rows	Share of raw logs
1–2 logs	859,079,266	974,119,684	19.4%
3–10 logs	426,997,616	2,320,754,180	46.3%
11–50 logs	57,239,096	980,142,273	19.5%
51+ logs	3,490,260	738,703,789	14.7%
Total	1,346,806,238	5,013,719,926	100.0%

5.2 Weak-label sources and benchmark scale

The supervised benchmark is built from two high-precision on-chain label sources. The first is decentralized-exchange trade activity from `crosschain.dex.trades`, which supplies labels for economic event classification. The second is lending-liquidation activity from `ethereum.lending.liquidations`, which supplies labels for liquidation detection. These labels are not treated as a complete taxonomy of all Ethereum behavior. They define task-labelled transaction instances for evaluating whether trace-native representations transfer across time, protocols, and contracts.

The labelled benchmark contains 311.97 million task-labelled transaction instances and 2.33 billion matched raw log rows. DEX event classification is the main large-scale task, with 311.87 million labelled transactions and 2.33 billion matched logs. Liquidation detection is the rare-event stress test, with 105,952 labelled transactions and 2.45 million matched logs.

Table 8: Task-labelled benchmark scale

Task	Label source	Labelled tx	Matched logs	Projects
DEX event classification	DEX swaps/trades	311.87M	2.33B	55
Liquidation detection	Lending liquidations	105,952	2.45M	21
Total	–	311.97M	2.33B	–

5.3 Temporal splits and raw-log coverage

The main evaluation uses a forward split. Training uses 2021–2024, validation uses January–June 2025, and testing uses July–December 2025. This design prevents future protocol deployments, future contracts, and future event vocabularies from entering the supervised training period. The test set is also mechanically harder than the training set: DEX test transactions have more logs, more unique event topics, and more emitting contracts per transaction than training transactions.

Table 9: Final benchmark split summary

Task	Split	Tx	Projects	Logs	Logs/tx	Topics/tx	Emitters/tx
DEX classification	Train 2021–2024	233.41M	50	1.65B	7.05	4.29	3.75
DEX classification	Validation 2025H1	37.38M	53	300.84M	8.05	4.49	4.20
DEX classification	Test 2025H2	41.08M	53	383.02M	9.32	4.73	4.53
Liquidation detection	Train 2021–2024	69,609	19	1.50M	21.55	9.77	7.68
Liquidation detection	Validation 2025H1	23,264	17	561,222	24.12	9.89	8.77
Liquidation detection	Test 2025H2	13,079	16	389,513	29.78	10.55	9.41

The raw-log join is nearly complete in the 2025H2 test period. For DEX-labelled test transactions, 41,061,915 of 41,076,269 transactions have at least one matched raw log, leaving 14,354 zero-log joins, or 0.035% of the split. All 13,079 liquidation-labelled test transactions have matched raw logs. The benchmark therefore retains the full labelled split while reporting raw-log coverage as a data-quality diagnostic.

5.4 Protocol, contract, and pool-infrastructure holdout

The benchmark contains real protocol-transfer and contract-transfer settings. Several protocol/project combinations are absent from the 2021–2024 training period but appear in the 2025 validation and test windows. The largest cases are Uniswap v4 and Ekubo v1. Uniswap v4 has no training-period transactions in the label manifest, then 1,851,562 validation transactions and 6,600,656 test transactions. Ekubo v1 has no training-period transactions, then 284,549 validation transactions and 1,972,900 test transactions. Together with smaller new projects, DEX protocols that first appear in validation and continue into test account for 8,618,559 DEX test transactions.

The pool or liquidity-infrastructure holdout is even stronger. Among DEX-labelled transactions with liquidity-pool identifiers, 8,674 identifiers first appear in validation and remain in the test period, accounting for 13.4 million test transactions. A further 55,559 identifiers appear only in the test period, accounting for 12.8 million test transactions. In total, 26.2 million DEX test transactions occur on pool or liquidity-infrastructure identifiers absent from the 2021–2024 training period. This design tests whether representations capture reusable event-stream structure rather than simply memorizing historical pool addresses.

Table 10: DEX pool or liquidity-infrastructure holdout

Holdout status	Identifiers	Validation tx	Test tx
Seen in train and test	55,167	34.0M	32.8M
New in validation, seen in test	8,674	12.0M	13.4M
Test-only identifier	55,559	0	12.8M
Train only or not in test	333,218	0.39M	0
Other	39,379	3.45M	0

5.5 Shortcut-risk diagnostics

Blockchain traces contain common identifiers and common event topics. The benchmark therefore reports shortcut diagnostics rather than assuming that all predictive signal reflects deep transaction structure. After excluding common token contracts such as WETH, USDC, USDT, DAI, WBTC, stETH, and wstETH, the most frequent remaining event topic in the 2025H2 DEX test split appears in 80.7% of DEX-labelled test transactions. For liquidation, the corresponding top-topic share is 95.2%. The top topic is the standard ERC-20 `Transfer` event signature, which shows that a naive classifier can exploit generic token-transfer evidence.

The diagnostic is then repeated after excluding generic Transfer, Approval, common Swap, common Sync, and other high-frequency generic topics. For DEX, the top-topic share falls to 14.35%, and the remaining event stream spans 6,783 unique topics and 97,084 emitting contracts. For liquidation, concentration remains higher: the top remaining topic appears in 53.53% of liquidation test transactions, across 434 remaining topics

and 1,797 emitting contracts. The paper therefore treats DEX event classification as the main large-scale representation-learning benchmark and liquidation detection as a rare-event stress test requiring stronger address- and topic-masked robustness checks.

Table 11: Event-topic shortcut diagnostics in the 2025H2 test split

Label	Top-topic before	Top-topic after	Topics after	Contracts after
DEX swap/trade	80.7%	14.35%	6,783	97,084
Liquidation	95.2%	53.53%	434	1,797

5.6 Leakage controls and reproducibility

All splits are time-forward and non-overlapping. Project, protocol, pool, and liquidity-infrastructure identifiers are recorded with first-seen and last-seen timestamps. Contract-holdout and topic-masked evaluations remove direct identifier or event-signature shortcuts from the supervised model input. The reproducibility package records the query manifests used to construct the benchmark, including the raw-log corpus manifest, complexity-bin manifest, protocol coverage manifest, weak-label manifest, temporal label manifest, protocol-holdout manifest, pool-holdout manifest, event-topic diagnostics, generic-topic-exclusion diagnostics, final benchmark split summary, and zero-log coverage diagnostic.

Benchmark leakage is treated as a first-order benchmark-design risk. EarthquakeNPP motivates its benchmark partly by showing that earlier earthquake datasets used non-sequential splits that created leakage across adjacent windows; ETH-TraceBench adopts the same principle for blockchain traces by separating label construction, input construction, split assignment, and model reporting. Table 12 records the specific leakage pathways considered in the current release.

Table 12: Leakage and shortcut-control safeguards in ETH-TraceBench

Risk	Safeguard	Evidence or required reporting rule
Temporal leakage	Train, validation, and test windows are strictly chronological: 2021–2024, 2025H1, and 2025H2.	Future protocols, pools, and event patterns cannot enter supervised training through random splits.
Transaction duplication across splits	Split assignment is based on transaction timestamp, and transaction hashes are treated as unique evaluation instances within each task label.	Split-manifest checks must report transaction counts by task and split before model tables are regenerated.
Protocol or pool memorization	Protocol-holdout and pool/liquidity-infrastructure holdout manifests are constructed separately from the main temporal split.	DEX includes 2025 protocols absent from training and 26.2M test transactions on pool or liquidity identifiers absent from the 2021–2024 training period.
Feature-label leakage	Decoded label-source fields such as project, protocol, and liquidation-table metadata are used to construct labels and diagnostics, not as supervised model inputs.	The reference input view is the raw event stream: emitting contracts, event topics, ordering, depth, and transaction-level metadata.
Address and topic shortcut learning	Address-masked, topic-masked, and address+topic-masked views are required for leaderboard-style claims.	Shortcut diagnostics report concentration before and after common-token and generic-topic exclusions.
Test-set overfitting	The 2025H2 test split is reserved for final reporting; model selection should use training and 2025H1 validation only.	Developer guidelines require seed-level prediction files and prohibit tuning on final test metrics.
Unsupported performance claims	Numerical model comparisons must be generated from saved transaction-level predictions.	The submission withholds model-superiority claims unless the relevant prediction artifacts are available and verified.

5.7 Benchmark validity and data-realism assessment

Benchmark papers must show not only that a dataset is large, but also that its split, labels, and evaluation rules correspond to a meaningful deployment problem. Following this logic, ETH-TraceBench treats benchmark validity as an explicit design object rather than as an assumption. Table 13 summarizes the main threats to validity and the safeguards implemented in the current verified release.

Table 13: Benchmark validity and data-realism assessment

Validity dimension	Safeguard and verified evidence	Remaining limitation
Temporal realism	Training, validation, and testing are separated chronologically: 2021–2024, 2025H1, and 2025H2. This prevents future protocol deployments and future pool identifiers from entering supervised training.	The benchmark is retrospective rather than continuously prospective.
Label validity	Labels are restricted to high-precision DEX trade and lending-liquidation tables and are documented as task labels, not a complete taxonomy of Ethereum behavior.	Rare mechanisms and parser changes require manual audit before supporting fine-grained claims.
Deployment relevance	Protocol-transfer and pool/liquidity-infrastructure holdout manifests isolate examples absent from the training period, including new 2025 protocols and test-only pool identifiers.	New mechanisms outside DEX and liquidations require additional label sources.
Shortcut validity	Address concentration, topic concentration, common-token exclusion, and generic-topic-exclusion diagnostics are reported; masked views are required for leaderboard-style claims.	Without saved predictions, the current release cannot yet quantify model-level shortcut dependence.
Input completeness	Raw-log coverage is measured directly. In 2025H2, the zero-log share is 0.035% for DEX and 0% for liquidations.	Some unmatched rows may remain after upstream table changes or future data refreshes.
Reproducibility	Query runners use monthly chunks, row limits, split manifests, and verification scripts for model outputs.	A full public leaderboard additionally requires standardized loaders and released prediction artifacts.

The validity assessment defines the evidentiary scope of the submission. The release verifies benchmark construction, evaluation protocol, scale, forward shift, protocol transfer, contract holdout, shortcut risk, raw-log coverage, and the reported baseline layers. Model-performance claims are limited to tables that can be regenerated from transaction-level prediction files.

6 Tasks, Splits, and Evaluation Protocol

The evaluation is organized around four axes. First, the forward temporal split tests whether representations learned before 2025 remain useful on 2025H2 transaction structure. Second, protocol-transfer and pool-holdout evaluations test whether models generalize to protocol/project combinations and liquidity-infrastructure identifiers absent from training. Third, masking evaluations remove contract identifiers, event topics, or both to measure shortcut dependence. Fourth, label-efficiency evaluations test whether self-supervised pretraining improves performance when downstream labels are scarce. Table 15 summarizes these axes.

For a benchmark submission, the evaluation protocol is part of the contribution. We therefore distinguish *task labels* from *evaluation conditions*. DEX swap/trade classification and liquidation detection provide the labels. Temporal, protocol-transfer, pool-holdout, address-masked, topic-masked, address+topic-masked, and label-efficiency settings provide the conditions under which a model must be evaluated.

Table 14: Recommended ETH-TraceBench leaderboard tasks

Leaderboard task	Label family	Evaluation subset
DEX temporal classification	DEX swap/trade	Full 2025H2 test split
DEX protocol transfer	DEX swap/trade	Protocol/project combinations absent from 2021–2024 training
DEX pool holdout	DEX swap/trade	Pool or liquidity-infrastructure identifiers absent from training
DEX address-masked	DEX swap/trade	Full test split with contract/address identifiers masked
DEX topic-masked	DEX swap/trade	Full test split with event topics masked or grouped
DEX address+topic masked	DEX swap/trade	Full test split with both shortcut families masked
Liquidation rare-event detection	Lending liquidations	Full 2025H2 liquidation test split and shortcut-controlled variants

Table 15: Evaluation axes for trace-native representation learning

Evaluation axis	Purpose
Forward temporal test	Measures performance when models train on 2021–2024 and test on 2025H2.
Protocol-transfer test	Measures performance on protocols or project/protocol combinations not present in training.
Pool or liquidity-infrastructure holdout	Measures performance on DEX test transactions whose pool or liquidity-infrastructure identifiers are absent from training.
Identifier and topic masking	Separates reusable trace structure from address, protocol-tag, and event-signature memorization.
Generic-topic exclusion	Measures residual event-stream diversity after common Transfer, Approval, Swap, and Sync topics are removed.
Label-efficiency test	Measures how much labelled data is needed after trace-native pretraining.

6.1 Baselines and model families

The benchmark protocol distinguishes *required baseline families* from the *verified baseline suite* included in this version. The present numerical suite is intentionally shallow: it uses only raw log count, unique event-topic count, and unique emitting-contract count to train dummy classifiers, a linear classifier, a random forest, and a gradient-boosted model. This suite establishes a lower-bound diagnostic for how much label information is carried by aggregate execution complexity alone.

A required benchmark layer is an Ethereum-native reference baseline. The natural analogue of a domain baseline is not another generic neural sequence model, but a topic/emitter rule or hashed topic-emitter model that uses the public event vocabulary and emitting-contract structure directly. Such baselines test whether DEX and liquidation labels are already recoverable from stable event signatures, contract emitters, or protocol parsers. MEM-BL and other trace-native models must be compared against these topic/emitter baselines under the full temporal split, protocol-transfer subsets, pool-holdout subsets, and identifier-masked views. Without this comparison, a neural model can appear strong while learning the same public identifiers as a simpler blockchain-native classifier.

The verified topic/emitter baseline suite reports this domain-native baseline layer numerically. The protocol extracts three symbolic transaction views from raw logs: an ordered topic bag, an ordered emitter-contract bag, and an ordered topic–emitter pair bag. These views deliberately expose the information that a shortcut-heavy Ethereum classifier is most likely to exploit. The resulting baselines are interpreted as stress tests rather than trace-semantic models: if a topic/emitter model performs as well as a sequence model, the sequence model has not demonstrated transferable event-stream structure.

Final submissions should additionally compare MEM-BL with a rule-based parser, hand-engineered random forest, XGBoost with protocol features, DeepLog-style sequence learning, LogBERT-style masked log modeling, LSTM, supervised Transformer, graph neural network over address-contract-transfer graphs, Temporal Graph Network, TGAT-style temporal attention, GraphMixer-style temporal-link encoder, generic masked-

Table 16: Ethereum-native topic/emitter baseline protocol

Baseline	Input view	Benchmark question
TopicBag-SGD	Hashed bag of raw `topic0` event signatures	Are task labels recoverable from stable event signatures without emitter identity or sequence modeling?
EmitterBag-SGD	Hashed bag of emitting contract addresses	Are task labels recoverable from contract identity alone, indicating address shortcut risk?
TopicEmitterHash-SGD	Hashed topic, emitter, and topic–emitter pair tokens	Does the public event vocabulary plus contract identity solve the task under the forward split?
TopicEmitterTrace-SGD	Topic/emitter hashes plus aggregate trace-count features	How much do the verified trace-count shortcuts add to Ethereum-native symbolic features?

token Transformer, and contrastive transaction model. These families represent parser-based, feature-based, sequence-based, graph-based, and self-supervised approaches that a benchmark user would reasonably try.

When regenerated, neural baselines should use the same temporal partitions, labels, optimizer family, early-stopping rule, validation metric, and at least three fixed random seeds. Transformer and generic masked-token baselines should be matched to MEM-BL on hidden size, attention heads, layer count where applicable, and pretraining budget. Sections A.2 and A.3 define the reporting fields needed to verify this comparability.

6.2 Minimum benchmark-validation suite

A benchmark paper should make clear which empirical claims can be supported by which baseline families. We therefore separate the minimum validation suite from the optional model-development space. Table 17 lists the baseline families that should be regenerated before any numerical performance claims are reintroduced. The suite is designed to answer four distinct questions: whether the labels are recoverable by trivial public-ledger shortcuts, whether shallow event-count models already solve the task, whether generic sequence or graph models generalize under the same splits, and whether trace-native pretraining adds value beyond those baselines.

Table 17: Minimum baseline suite required before reintroducing model-performance claims

Baseline family	What it tests	Required evidence before reporting results
Shortcut-only classifier	Whether addresses, protocol tags, or top event topics recover labels without trace structure	Full, address-masked, topic-masked, and address+topic-masked scores; feature list used by the shortcut model
Rule/parser baseline	Whether deterministic event signatures already solve the benchmark	Rule definitions, topic/address lists, failure cases, and temporal, protocol-transfer, and pool-holdout results
Bag-of-events linear model	Whether unordered topic, emitter, token, gas, or log-depth counts are sufficient	Verified for TopicBag-SGD, EmitterBag-SGD, TopicEmitterHash-SGD, and TopicEmitterTrace-SGD across three seeds, two tasks, and validation/test splits
Tree or boosted-tabular model	Whether engineered transaction and protocol features explain labels without sequence modeling	Feature groups, validation protocol, hyperparameters, and holdout degradation relative to the temporal test
Generic sequence model	Whether standard log-sequence learning captures event ordering without blockchain-specific pretraining	Tokenization, truncation rule, pretraining budget if any, and matched splits
Graph or temporal-graph model	Whether address/contract-transfer structure outperforms transaction-internal event streams	Graph construction window, node/edge features, leakage controls, and protocol/pool-holdout results
Trace-native reference model	Whether masked event-stream pretraining improves over generic sequence and graph alternatives	Saved per-seed predictions, configuration hashes, ablations, and paired tests against the strongest baseline

Table 17 defines the reporting requirement for a full leaderboard. The verified trace-count, topic/emitter, and lightweight neural baseline suites instantiate the artifact-backed rows reported in the manuscript. Parser, sequence, graph, generic Transformer, or trace-native model claims belong in the table only when each row is backed by the same prediction schema. The requirement keeps model evidence tied to auditable artifacts rather than unverifiable leaderboard entries.

6.3 Metric rationale and statistical testing

A benchmark for DeFi event streams should not be evaluated by accuracy alone. DEX activity is large and heterogeneous, while liquidation is a rare-event stress test. A model that performs well on frequent, easy cases can therefore look strong under aggregate accuracy while failing on the minority mechanism that matters for monitoring. We specify the metric suite before reporting any model results so that future submissions are comparable and so that the benchmark does not reward shortcut-heavy classifiers. Table 18 defines the required metrics and the role of each metric in ETH-TraceBench.

Table 18: Metric rationale for ETH-TraceBench evaluation

Metric	Primary use	Rationale in Ethereum DeFi event streams
Macro-F1	Primary leaderboard metric	Weights classes more evenly than accuracy and penalizes models that recover only the dominant or easiest class. This is the default summary metric for DEX and shortcut-controlled variants.
AUPRC	Rare-event and imbalanced settings	More informative than AUROC when the positive class is rare; required for liquidation and other sparse monitoring labels.
AUROC	Ranking quality	Measures whether scores rank positives above negatives across thresholds, but should be reported with AUPRC rather than alone.
Rare-class recall	Monitoring sensitivity	Captures whether a model retrieves uncommon mechanisms such as liquidation-related transactions, where false negatives may be costly.
Expected calibration error	Decision support	Measures whether prediction scores can be used as confidence estimates for triage thresholds or downstream alerting.
Shift degradation	Generalization under holdout	Reports the difference between full temporal-test performance and protocol-transfer, pool-holdout, address-masked, topic-masked, or address+topic-masked performance.
Runtime and memory	Ethereum-scale feasibility	Required because a model that is accurate on small samples may be impractical for hundreds of millions of transactions and billions of raw logs.

Leaderboard comparisons should report 95% bootstrap confidence intervals over the same test transactions. Primary comparisons should use paired bootstrap tests because competing models are evaluated on the same transaction instances, and multiple comparisons across the main task family should be adjusted with the Holm procedure. Model reports should include the number of seeds, the random seeds used, the validation criterion, the test condition, and the prediction-artifact hash for each result table. Numerical claims in this manuscript are restricted to the verified trace-feature, linear topic/emitter, lightweight neural topic/emitter, and slice-audit artifacts; masking, holdout, ablation, and runtime claims remain part of the required reporting schema.

6.4 Reproducibility protocol

All benchmark-diagnostic tables in this version are generated from extraction manifests and query outputs rather than manual spreadsheet entries. The final model-performance package should additionally contain experiment configurations, saved per-seed predictions, table-generation code, transaction-hash lists for su-

pervised benchmarks, and small smoke-test subsets that can be reproduced without a full blockchain mirror. Each final manifest should record a data snapshot hash, split hash, label hash, model configuration, random seed, training budget, software version, and evaluation-script hash. Raw public blockchain data can be reconstructed from the documented extraction pipeline when full mirrors cannot be redistributed.

7 Verified Benchmark Diagnostics

The results section reports verified benchmark diagnostics, trace-count baselines, Ethereum-native topic/emitter baselines, a lightweight neural topic/emitter baseline, and slice-level shortcut audits. The diagnostics evaluate whether the benchmark is a meaningful representation-learning test rather than an in-domain replay of known contracts and event names. Because shortcut learning is a central risk in blockchain traces, temporal-shift, protocol-transfer, pool-holdout, topic-concentration, generic-topic-exclusion, and raw-log-coverage evidence are reported before the baseline results are interpreted.

7.1 Benchmark diagnostics and shortcut tests

The revised benchmark construction produces two important diagnostic results before model performance is interpreted. First, the 2025H2 test split is not simply a random sample from the training distribution. DEX test transactions are larger and more compositionally complex than train transactions: average logs per transaction rise from 7.05 in training to 9.32 in test, and average emitting contracts per transaction rise from 3.75 to 4.53. The pattern makes the forward test split a temporal distribution-shift benchmark rather than an in-domain replay of historical transaction structure.

Second, the test split contains substantial protocol and contract shift. Protocol/project combinations absent from the 2021–2024 training period account for 8.62 million DEX test transactions. Pool or liquidity-infrastructure identifiers absent from the training period account for 26.2 million DEX test transactions. The resulting contract-holdout setting prevents evaluation from relying only on historical pool-address memorization. Figure 2 summarizes these difficulty diagnostics in the same visual spirit as benchmark papers that first establish the benchmark object before reporting model comparisons.

Table 19: Forward-shift diagnostics for the DEX benchmark

Diagnostic	Train 2021–2024	Test 2025H2
DEX labelled transactions	233.41M	41.08M
Raw logs matched to DEX labels	1.65B	383.02M
Average logs per transaction	7.05	9.32
Average unique topics per transaction	4.29	4.73
Average emitting contracts per transaction	3.75	4.53
Projects/protocols	50	53

Shortcut diagnostics are equally important. Before generic-topic exclusions, the top event topic appears in 80.7% of DEX test transactions and 95.2% of liquidation test transactions. The top topic is the ERC-20 Transfer signature, which is common across many economically distinct transactions. After excluding common token contracts and generic Transfer, Approval, Swap, and Sync topics, DEX concentration falls sharply: the top remaining topic appears in 14.35% of DEX test transactions, and the residual event stream spans 6,783 event topics and 97,084 emitting contracts. Liquidation remains more concentrated, with a top remaining topic share of 53.53%. We therefore interpret DEX classification as the main large-scale representation-learning benchmark and liquidation detection as a rare-event stress test.

Table 20: Shortcut diagnostics after common-contract and generic-topic exclusions

Label	Test tx after exclusions	Unique topics	Unique contracts	Top-topic tx share
DEX swap/trade	18.08M	6,783	97,084	14.35%
Liquidation	13,079	434	1,797	53.53%

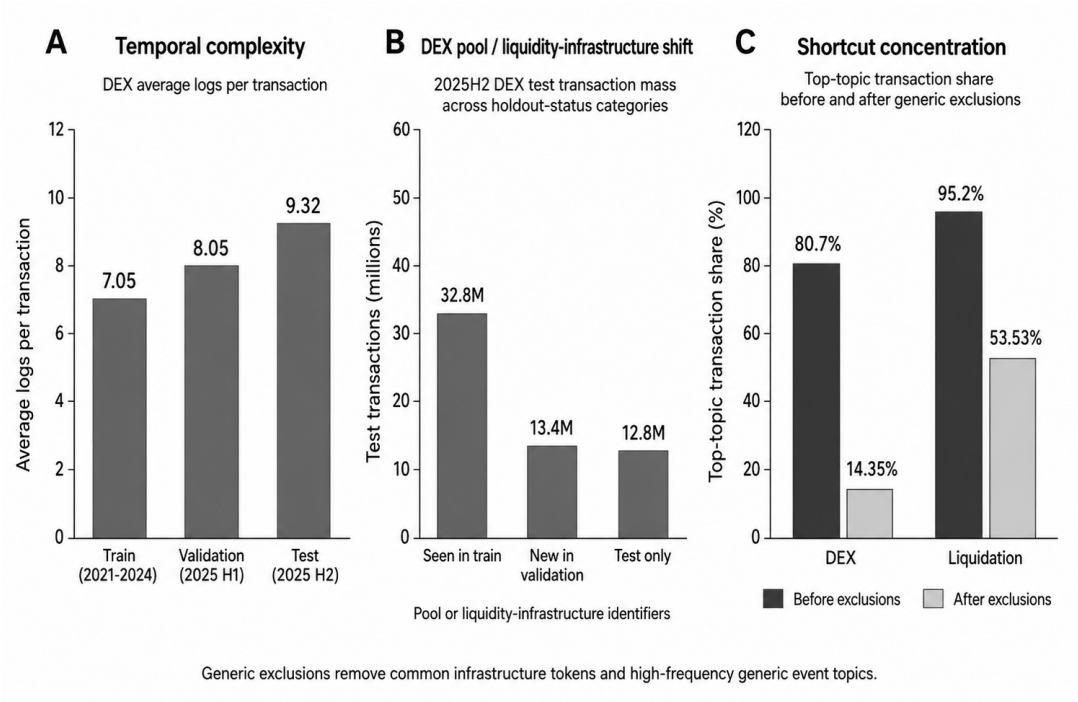

Figure 2: Benchmark difficulty diagnostics. Panel A shows that the DEX test split contains richer traces than the training split. Panel B shows that 2025H2 contains substantial liquidity-infrastructure shift, including identifiers absent from the 2021–2024 training window. Panel C shows why shortcut controls are necessary: generic event topics are highly predictive before exclusions, but DEX top-topic concentration falls sharply after excluding common contracts and generic event topics.

The diagnostics define the conditions under which downstream model results are interpreted. Positive results under the full feature set are not sufficient evidence of transferable trace understanding. Protocol-transfer, pool-holdout, address-masked, topic-masked, and address+topic-masked evaluations provide the more informative comparisons because they reduce the value of direct memorization.

7.2 Verified numeric trace-feature baseline suite

We next report a limited baseline suite that can be regenerated from transaction-level prediction files. These baseline experiments are not run over the full 311.97 million labelled benchmark instances. Instead, they use a fixed 911,291-row supervised evaluation sample produced from the warehouse extraction after removing the 400-row smoke-test chunks. The DEX task contributes 670,140 rows, and the liquidation task contributes 241,151 rows. Training uses the 2021–2024 portion of this sample; validation and test metrics use fixed 2025H1 and 2025H2 evaluation slices. The DEX validation and test slices are class-balanced at 85,000 transactions each, while the liquidation 2025H2 slice preserves the rare-event test imbalance. This design makes the baseline tables reproducible and lightweight, while the full-scale benchmark counts, split manifests, and holdout manifests define the larger evaluation universe for future leaderboard submissions. The prediction archive contains 60 files: five models, three random seeds, two tasks, and two evaluation splits.

The baseline feature set is deliberately minimal. Each transaction is represented only by raw log count, unique event-topic count, and unique emitting-contract count. The models are Dummy-Majority, Dummy-Stratified, TraceStats-LR, TraceStats-RF, and TraceStats-GB. The purpose is not to establish state-of-the-art model performance; it is to quantify how much signal is available from shallow execution-complexity statistics before sequence order, token-flow direction, contract identifiers, or graph context are introduced.

Table 21: Verified numeric trace-feature baselines

Task	Split	Model	N	Pos. rate	Macro-F1	Pos.-F1	AUROC	AUPRC
DEX	Val. 2025H1	Dummy-Majority	85,000	0.500	0.333	0.667	0.500	0.500
DEX	Val. 2025H1	Dummy-Stratified	85,000	0.500	0.495	0.543	0.500	0.500
DEX	Val. 2025H1	TraceStats-LR	85,000	0.500	0.859	0.857	0.942	0.910
DEX	Val. 2025H1	TraceStats-RF	85,000	0.500	0.959	0.960	0.978	0.963
DEX	Val. 2025H1	TraceStats-GB	85,000	0.500	0.964	0.964	0.981	0.968
DEX	Test 2025H2	Dummy-Majority	85,000	0.500	0.333	0.667	0.500	0.500
DEX	Test 2025H2	Dummy-Stratified	85,000	0.500	0.495	0.543	0.500	0.500
DEX	Test 2025H2	TraceStats-LR	85,000	0.500	0.863	0.860	0.943	0.910
DEX	Test 2025H2	TraceStats-RF	85,000	0.500	0.950	0.951	0.971	0.953
DEX	Test 2025H2	TraceStats-GB	85,000	0.500	0.953	0.954	0.975	0.960
Liquidation	Val. 2025H1	Dummy-Majority	46,528	0.500	0.333	0.667	0.500	0.500
Liquidation	Val. 2025H1	Dummy-Stratified	46,528	0.500	0.500	0.516	0.500	0.500
Liquidation	Val. 2025H1	TraceStats-LR	46,528	0.500	0.922	0.921	0.982	0.974
Liquidation	Val. 2025H1	TraceStats-RF	46,528	0.500	0.929	0.928	0.988	0.984
Liquidation	Val. 2025H1	TraceStats-GB	46,528	0.500	0.930	0.929	0.988	0.985
Liquidation	Test 2025H2	Dummy-Majority	143,869	0.091	0.083	0.167	0.500	0.091
Liquidation	Test 2025H2	Dummy-Stratified	143,869	0.091	0.387	0.156	0.501	0.091
Liquidation	Test 2025H2	TraceStats-LR	143,869	0.091	0.828	0.699	0.983	0.793
Liquidation	Test 2025H2	TraceStats-RF	143,869	0.091	0.842	0.722	0.986	0.831
Liquidation	Test 2025H2	TraceStats-GB	143,869	0.091	0.845	0.728	0.985	0.823

Notes: Metrics are means over three random seeds. The verified prediction archive contains 60 prediction files: five models × three seeds × two tasks × two evaluation splits. Models use only three aggregate transaction-trace features: raw log count, unique event-topic count, and unique emitting-contract count. These baselines diagnose shallow execution-complexity signal; they are not evidence that a model has learned transferable event semantics.

The results show that aggregate execution complexity is a strong signal. On DEX classification, TraceStats-GB reaches 0.953 macro-F1 and 0.960 AUPRC on the 2025H2 test split. On liquidation detection, TraceStats-GB reaches 0.845 macro-F1 on the imbalanced 2025H2 test split, while TraceStats-RF obtains the highest liquidation-test AUPRC at 0.831. These values should be interpreted as benchmark diagnostics rather than evidence of transferable event semantics. Because the baselines use only three aggregate trace counts, their strength indicates that many labels are partly recoverable from coarse execution complexity. This finding strengthens the case for protocol-transfer, pool-holdout, address-masked, topic-masked, and address+topic-masked evaluation before claiming that a richer model has learned reusable trace structure.

7.3 Verified Ethereum-native topic/emitter baseline suite

We then report the Ethereum-native symbolic baseline layer on a separate fixed topic/emitter sample. This suite uses the topic/emitter feature extraction protocol described above and an 869,147-row sample regenerated from raw logs after the smoke-test chunks were removed. As with the trace-count suite, these are reproducible baseline samples rather than a claim that all full-scale benchmark instances have been scored. The prediction archive contains 48 files: four models, three random seeds, two tasks, and two evaluation splits. The models are TopicBag-SGD, EmitterBag-SGD, TopicEmitterHash-SGD, and TopicEmitterTrace-SGD. The first three use hashed symbolic views of event topics, emitter addresses, and topic–emitter pairs; the fourth adds the three aggregate trace-count features used in Table 21.

The topic/emitter results sharpen the interpretation of the benchmark. For DEX classification, topic/emitter baselines are strong but do not dominate the best trace-count gradient-boosted model on 2025H2 macro-F1: TopicEmitterTrace-SGD reaches 0.924 macro-F1, while TraceStats-GB reaches 0.953. However, TopicBag-SGD obtains the highest DEX test AUPRC among the symbolic models at 0.978, showing that public event signatures remain highly informative. For liquidation detection, the Ethereum-native baselines are much stronger than the trace-count suite: TopicEmitterTrace-SGD reaches 0.943 macro-F1 and 0.939 AUPRC on the imbalanced 2025H2 test split, while TopicBag-SGD obtains 0.965 AUPRC. This confirms that liquidation labels are especially vulnerable to event-signature and emitter shortcuts. Consequently, liquidation should be used as a rare-event stress test with stronger masking and holdout controls, rather than as evidence that a model has learned general trace semantics.

We also add a lightweight neural baseline, TopicEmitterHashMLP, that embeds hashed event topics, emitters, and topic–emitter tokens, mean-pools the transaction-level symbolic set, and trains a small multi-layer perceptron on the supervised task labels. This model is intentionally weaker than the full MEM-BL

Table 22: Verified Ethereum-native topic/emitter baselines

Task	Split	Model	N	Pos. rate	Macro-F1	Pos.-F1	AUROC	AUPRC
DEX	Val. 2025H1	TopicBag-SGD	85,000	0.500	0.973	0.972	0.992	0.994
DEX	Val. 2025H1	EmitterBag-SGD	85,000	0.500	0.928	0.926	0.958	0.925
DEX	Val. 2025H1	TEHash-SGD	85,000	0.500	0.933	0.931	0.97	0.954
DEX	Val. 2025H1	TETrace-SGD	85,000	0.500	0.949	0.949	0.982	0.973
DEX	Test 2025H2	TopicBag-SGD	85,000	0.500	0.918	0.912	0.966	0.978
DEX	Test 2025H2	EmitterBag-SGD	85,000	0.500	0.900	0.894	0.946	0.924
DEX	Test 2025H2	TEHash-SGD	85,000	0.500	0.899	0.891	0.954	0.939
DEX	Test 2025H2	TETrace-SGD	85,000	0.500	0.924	0.921	0.975	0.967
Liquidation	Val. 2025H1	TopicBag-SGD	46,528	0.500	0.984	0.984	0.999	0.998
Liquidation	Val. 2025H1	EmitterBag-SGD	46,528	0.500	0.981	0.981	0.995	0.993
Liquidation	Val. 2025H1	TEHash-SGD	46,528	0.500	0.983	0.983	0.998	0.998
Liquidation	Val. 2025H1	TETrace-SGD	46,528	0.500	0.992	0.993	0.998	0.997
Liquidation	Test 2025H2	TopicBag-SGD	143,861	0.091	0.942	0.895	0.996	0.965
Liquidation	Test 2025H2	EmitterBag-SGD	143,861	0.091	0.937	0.886	0.983	0.899
Liquidation	Test 2025H2	TEHash-SGD	143,861	0.091	0.943	0.895	0.986	0.945
Liquidation	Test 2025H2	TETrace-SGD	143,861	0.091	0.943	0.897	0.996	0.939

Notes: Metrics are means over three random seeds. Full standard deviations and per-seed metrics are archived in the topic/emitter result tables. TEHash-SGD abbreviates TopicEmitterHash-SGD; TETrace-SGD abbreviates TopicEmitterTrace-SGD. These baselines deliberately expose Ethereum event signatures and emitter-contract identities and should therefore be interpreted as shortcut and parser-proximity stress tests rather than trace-semantic models.

reference protocol: it does not pretrain, does not model ordered nested calls, and does not use typed flow objectives. Its purpose is to anchor the benchmark against a simple neural model that can exploit Ethereum-native symbolic structure while remaining fully reproducible from saved predictions.

Table 23: Verified lightweight neural topic/emitter baseline

Task	Split	Model	N	Pos. rate	Macro-F1	Pos.-F1	AUROC	AUPRC
DEX	Val. 2025H1	TopicEmitterHashMLP	85,000	0.500	0.985 ± 0.000	0.985 ± 0.000	0.999 ± 0.000	0.999 ± 0.000
DEX	Test 2025H2	TopicEmitterHashMLP	85,000	0.500	0.959 ± 0.000	0.959 ± 0.000	0.994 ± 0.000	0.994 ± 0.000
Liquidation	Val. 2025H1	TopicEmitterHashMLP	46,528	0.500	0.993 ± 0.001	0.993 ± 0.001	0.999 ± 0.000	0.999 ± 0.000
Liquidation	Test 2025H2	TopicEmitterHashMLP	143,861	0.091	0.938 ± 0.004	0.889 ± 0.007	0.998 ± 0.000	0.965 ± 0.003

Notes: Metrics are means ± standard deviations over three random seeds. The model uses hashed topic, emitter, and topic–emitter tokens with mean pooling and a small MLP classifier. It is a lightweight neural symbolic baseline, not a MEM-BL model-superiority result. The result is included because benchmark papers should compare against simple neural baselines as well as shallow and linear symbolic baselines.

The neural baseline strengthens the benchmark evidence in two ways. First, it shows that the evaluation is not only a contest among shallow linear models: a small neural topic/emitter encoder improves DEX test macro-F1 to 0.959, slightly above the strongest trace-count model and above the linear topic/emitter suite. Second, it preserves the main benchmark warning. The same model drops from 0.985 to 0.959 macro-F1 under the DEX forward split and from 0.993 to 0.938 macro-F1 under the liquidation forward split. For liquidation, the neural model obtains high AUPRC but does not remove the shortcut concern because it still consumes event-topic and emitter identifiers. Therefore, its results are best interpreted as an additional baseline layer, not as evidence that trace-native representation learning has solved the benchmark.

The verified baseline archives also allow a first forward-shift degradation calculation. Table 24 compares the strongest verified trace-count model, strongest linear topic/emitter model, and lightweight neural topic/emitter model on validation and test splits. This is not a substitute for protocol-transfer or pool-holdout evaluation, but it makes the reporting convention explicit: benchmark results should be read through performance gaps across deployment-relevant conditions, not only through the best aggregate score.

The DEX trace-count gap is small under the full forward split, but this score still uses only aggregate complexity and does not isolate protocol-transfer or pool-holdout subsets. The linear and neural topic/emitter DEX gaps are larger, suggesting that event-signature shortcuts transfer less perfectly to 2025H2 than trace-count shortcuts. The liquidation topic/emitter baselines remain very strong despite the forward split, which reinforces the need for topic-masked and contract-holdout evaluations for that task. These results define the reporting convention used throughout the benchmark: report the full temporal score and the degradation under each OOD or shortcut-controlled condition.

Table 24: Verified forward-shift degradation for strongest reported baseline layers

Task	Model	Val. MF1	Test MF1	Gap	Test AUPRC
DEX event classification	TraceStats-GB	0.964	0.953	−0.011	0.960
DEX event classification	TopicEmitterTrace-SGD	0.949	0.924	−0.025	0.967
DEX event classification	TopicEmitterHashMLP	0.985	0.959	−0.026	0.994
Liquidation detection	TraceStats-GB	0.930	0.845	−0.085	0.823
Liquidation detection	TopicEmitterTrace-SGD	0.992	0.943	−0.049	0.939
Liquidation detection	TopicEmitterHashMLP	0.993	0.938	−0.055	0.965

7.4 Verified OOD slice and shortcut audit

We add a slice-level audit of the verified topic/emitter prediction archive. For each task, the audit constructs training-set vocabularies for event topics, emitter contracts, and topic–emitter pairs. It then partitions validation and test predictions into examples whose symbolic elements were fully seen in the 2021–2024 training split and examples containing at least one unseen symbolic element. This is a stronger shortcut diagnostic than the aggregate forward split because it asks whether a domain-native classifier transfers beyond the exact public event signatures or topic–emitter pairs it observed during training.

Table 25 reports the 2025H2 test slices for the two strongest symbolic baselines. The DEX task shows the clearest shortcut-transfer gap. For TopicBag-SGD, macro-F1 falls from 0.977 on all-seen topics to 0.655 on transactions containing unseen topics, and from 0.983 to 0.834 when moving from all-seen topic–emitter pairs to transactions containing unseen pairs. TopicEmitterTrace-SGD is less brittle but still degrades from 0.950 to 0.772 on unseen-topic DEX transactions and from 0.960 to 0.860 on unseen topic–emitter-pair transactions. Liquidation remains easier for symbolic baselines, but even there TopicEmitterTrace-SGD falls from 0.989 to 0.908 on unseen topic–emitter pairs. These results confirm that public event signatures and contract emitters are not merely useful features; they are also a measurable source of shortcut risk.

Table 25: Verified OOD slice audit for topic and topic–emitter shortcut transfer on the 2025H2 test split

Task	Model	Slice	Seen N	Unseen N	Seen MF1	Unseen MF1	Gap	Seen AUPRC	Unseen AUPRC
DEX	TETrace-SGD	Topic	69,228	15,772	0.950	0.772	−0.178	0.975	0.948
DEX	TETrace-SGD	Topic–emitter pair	44,364	40,636	0.960	0.860	−0.100	0.965	0.970
DEX	TopicBag-SGD	Topic	69,228	15,772	0.977	0.655	−0.322	0.996	0.979
DEX	TopicBag-SGD	Topic–emitter pair	44,364	40,636	0.983	0.834	−0.149	0.994	0.988
Liquidation	TETrace-SGD	Topic	120,455	23,406	0.973	0.878	−0.095	0.985	0.878
Liquidation	TETrace-SGD	Topic–emitter pair	85,879	57,982	0.989	0.908	−0.081	0.997	0.899
Liquidation	TopicBag-SGD	Topic	120,455	23,406	0.953	0.912	−0.041	0.968	0.975
Liquidation	TopicBag-SGD	Topic–emitter pair	85,879	57,982	0.971	0.917	−0.054	0.978	0.955

Notes: Slice membership is computed relative to the 2021–2024 training split. "Unseen" indicates at least one topic or topic–emitter pair absent from the task-specific training vocabulary. Gaps are unseen minus seen macro-F1. TETrace-SGD abbreviates TopicEmitterTrace-SGD. Full slice metrics are archived with the release artifacts. One-class trace-length slices are not emphasized because AUROC is undefined.

The slice audit changes the interpretation of the strong aggregate topic/emitter results. High full-test scores are compatible with substantial degradation on transactions containing unseen symbolic elements. Therefore, a trace-native neural model is not judged only against the aggregate TopicBag or TopicEmitterTrace score. It is also evaluated on the same seen/unseen topic, emitter, and topic–emitter slices, and it should report whether its gains are concentrated on symbolic novelty rather than on repeated parser-like signatures.

7.5 Prediction-artifact protocol for model tables

Model-performance tables are regenerated from saved predictions rather than entered manually. Each prediction file should contain at least the task name, split, evaluation condition, model name, random seed, transaction identifier, ground-truth label, predicted label, and predicted probability or score for the positive or rare class. The verified trace-feature and topic/emitter baseline archives satisfy this core schema for the forward validation and test splits. For holdout and masking tables, prediction files additionally record the holdout flag, project or protocol identifier, pool or liquidity-infrastructure identifier, and the masking condition used at inference time. The metric-generation script should then group predictions by task, split,

condition, model, and seed; compute macro-F1, accuracy, rare-class recall, AUPRC, false-positive rate where applicable, and calibration error; and report means with bootstrap confidence intervals over test transactions.

The verification rule is simple: no performance number appears in the manuscript unless it can be regenerated from a saved prediction artifact using the table-generation script. The current manuscript follows this rule for the numeric trace-feature, linear topic/emitter, and lightweight neural topic/emitter baselines. The rule is especially important for this benchmark because direct identifiers, event signatures, and project labels are predictive. A result on the full feature set is not sufficient evidence of transferable trace understanding unless the same model is also evaluated under the shortcut-controlled conditions in Table 15. Appendix Tables 26, 27, and 28 provide the detailed artifact checklist and table-regeneration schema.

7.6 Scope of reported model evidence

The reported results establish benchmark difficulty and shortcut risk, not trace-native model superiority. A MEM-BL or neural trace-encoder comparison table should use the same 2021–2024 training split, select checkpoints using 2025H1 validation macro-F1, evaluate on the 2025H2 test split, and then restrict the same prediction files to protocol-transfer and pool-holdout subsets. The same runs should also be repeated under address-masked, topic-masked, and address+topic-masked inputs and under fixed label-efficiency subsamples. Appendix Table 28 records the performance tables that should be regenerated from those artifacts before any model-superiority claim is made.

8 Benchmark Package and Release Interface

The benchmark-paper model places as much weight on the evaluation interface as on the raw dataset. Accordingly, ETH-TraceBench is organized as an auditable release rather than as a single static table dump. The verified package supports the benchmark-construction claims through SQL queries, monthly chunk outputs, split manifests, holdout manifests, shortcut diagnostics, metric-regeneration code, a transaction-level prediction archive for numeric trace-feature baselines, a verified linear Ethereum-native topic/emitter baseline archive, and a verified lightweight neural topic/emitter baseline archive. The leaderboard interface is specified through standardized loaders, masking/holdout prediction schemas, runtime logs, checkpoints, and baseline-family requirements so that new methods can be compared without reimplementing the data split or shortcut filters.

Appendix A contains the supplementary material cited from the main text. It includes operational extraction (Appendix A.4), weak-label rules (Appendix A.5), manual-audit templates (Appendix A.6), split manifests (Appendix A.7), shortcut diagnostics (Appendix A.8), objective-ablation requirements (Appendix A.9), representation diagnostics (Appendix A.9), runtime scaling (Appendix A.9), optional external-MEV matching (Appendix A.10), error analysis (Appendix A.11), responsible release (Appendix A.12), reproducibility (Appendix A.13), and repository structure (Appendix A.14). The corresponding appendix tables are Tables 26–32, 33–35, 36, 37, 38, 39, 40, 41, 42, 43, and 44.

The package components that form the benchmark contract are summarized in Appendix Table 29. The distinction between verified artifacts and specified leaderboard artifacts is intentional. Construction artifacts verify the scale, split design, shift manifests, and shortcut diagnostics reported in this paper. The trace-feature, linear topic/emitter, and neural topic/emitter prediction archives verify the reported baseline tables. Broader model claims require the extended prediction-artifact layer described in Section 7.5.

Appendix Table 30 adds the runtime and computational-cost record that should accompany a benchmark release. The current evidence verifies prediction-level results for the shallow, linear topic/emitter, and neural topic/emitter suites, but full leaderboard claims should also include hardware, wall-clock time, memory, and artifact hashes. This information is necessary because Ethereum-scale benchmarks can be made artificially attractive by evaluating only small extracts or by omitting extraction cost.

A minimal loader interface should make the benchmark difficult to misuse. In particular, the loader should expose temporal split membership, protocol-holdout flags, pool-holdout flags, and masking conditions as

first-class fields rather than leaving each developer to reconstruct them. Appendix Table 31 specifies the target interface for a complete release.

For responsible release, we recommend three levels of artifact availability. Level 1 releases construction manifests, split definitions, query templates, and diagnostic tables, which is the level supported by the verified evidence in this manuscript. Level 2 adds standardized loaders and sanitized benchmark instances so that models can be trained without rerunning warehouse queries. Level 3 adds baseline checkpoints, transaction-level predictions, table-generation logs, and leaderboard metadata. The present manuscript is a Level 1 benchmark-construction release with a verified Level 3 baseline layer for trace-count and topic/emitter models; the standardized Level 2 loader and broader Level 3 checkpoint interface are specified as leaderboard-interface requirements.

9 Developer Guidelines and Reproducibility Requirements

A benchmark is useful only if results are comparable. We therefore define reporting rules for method developers using ETH-TraceBench. First, report the full temporal test result and the shortcut-controlled variants separately; do not collapse unmasked, address-masked, topic-masked, and address+topic-masked scores into one number. Second, report protocol-transfer and pool-holdout scores as separate generalization results. Third, report macro-F1, AUROC or AUPRC where appropriate, rare-class recall, calibration error, runtime, and memory. Fourth, do not tune hyperparameters on the 2025H2 test split. Fifth, release transaction-level predictions or sanitized transaction keys so that metrics, confidence intervals, paired tests, and holdout subsets can be recomputed. Appendix Table 32 gives the detailed reporting checklist.

The reporting requirements mirror benchmark packages that standardize data loading and evaluation. They are especially important for public ledgers because direct identifiers and event signatures can be legitimate signals for some monitoring tasks but shortcuts for representation-learning claims. ETH-TraceBench therefore separates full-feature performance from shortcut-controlled performance rather than treating a single aggregate score as sufficient evidence of generalization.

10 Blockchain Monitoring Implications

The verified evidence in this version supports a benchmark-design claim and three baseline-layer claims: aggregate trace counts are strongly predictive, linear Ethereum-native topic/emitter signatures are strongly predictive, and a lightweight neural topic/emitter encoder is also highly competitive. Ethereum execution traces provide a high-scale data layer for DeFi monitoring because they preserve nested calls, typed events, token-flow structure, call depth, gas use, and block-local ordering. The evidence also shows why naive evaluation can be misleading. Common event topics and infrastructure contracts are highly predictive before masking, and liquidation labels remain more concentrated than DEX labels even after generic-topic exclusions. The verified trace-feature baselines add a further warning: even three aggregate execution-complexity features can achieve high forward-split performance. The verified linear and neural topic/emitter baselines show a complementary shortcut risk: symbolic event signatures and emitter identities are especially powerful for liquidation detection. The OOD slice audit strengthens this point by showing that symbolic baselines degrade on transactions containing unseen topics and unseen topic–emitter pairs, especially in DEX classification. A credible model result therefore requires evaluation under temporal shift, protocol transfer, pool or liquidity-infrastructure holdout, seen/unseen symbolic slices, and identifier or topic masking.

The distinction matters for practical blockchain monitoring. Parsers, registries, and curated DeFi tables remain necessary for high-precision attribution. Learned representations are most useful when evaluated as complements to those tools: they can prioritize uncertain cases, support monitoring under new protocol versions, and provide representations for label-scarce tasks. ETH-TraceBench makes such claims testable by requiring prediction artifacts, shortcut-controlled reruns, and automatically regenerated comparison tables for MEM-BL or any other trace-native model.

The evidence supports three benchmark implications. First, benchmark design should report raw event coverage, task-labelled coverage, temporal splits, and raw-log completeness rather than only aggregate transaction

counts. Second, distribution shift should be measured directly through protocol-transfer and pool-holdout manifests rather than assumed from chronological splits. Third, shortcut diagnostics should be part of the main evidence because public ledgers expose identifiers and event signatures that are informative but can overstate generalization.

11 Limitations and Responsible Release

The empirical setting is Ethereum mainnet, so the benchmark should be interpreted as evidence for EVM-style execution traces rather than all blockchains. The benchmark should transfer most directly to other EVM chains because they share logs, calls, topics, token standards, and smart-contract execution semantics, but non-EVM chains require adaptation because account models, instruction formats, transaction ordering, program interfaces, and event schemas differ. The framework also depends on observable execution traces, token-flow side effects, and partial ABI or topic availability. It is less suitable when the chain does not expose comparable event logs or when economic state changes are primarily off-chain.

The benchmark depends on careful data engineering, label validation, protocol registry construction, contract upgrade tracking, and transaction-state reconstruction. The verified supervised labels in this version come from curated DEX trade and lending-liquidation tables, not from a complete manual taxonomy of Ethereum behavior. Weak labels may encode table-construction assumptions, and rare or ambiguous mechanisms may require manual audit before they are used as performance evidence. The package contains saved predictions and regenerated tables for the numeric trace-feature, linear topic/emitter, and lightweight neural topic/emitter baselines. It does not report MEM-BL checkpoints, shortcut-controlled neural-model reruns, ablation predictions, full runtime logs, or leaderboard-ready model cards; those artifacts are required before any neural model-superiority claim is made.

Privacy and misuse risks remain. Public blockchain data can sometimes be linked to people, firms, or trading entities. Learned representations can support defensive monitoring but may also help strategic actors identify profitable ordering or liquidation opportunities. A responsible release should avoid unnecessary deanonymization, should not publish mappings from wallets to real-world identities, and should withhold or sanitize examples that could make active vulnerabilities easier to exploit. Public examples should be delayed or sanitized when they involve active vulnerabilities. Dataset documentation should state intended uses, excluded uses, known limitations, and risks.

12 Conclusions

The paper introduced ETH-TraceBench, a large-scale benchmark for Ethereum DeFi event streams under naturally occurring temporal, protocol, contract, and topic-distribution shift. The benchmark is motivated by a gap in existing blockchain and machine-learning evaluation. Ethereum traces are public and high scale, but their economic meaning emerges from nested calls, typed logs, token flows, call depth, gas use, and block-local ordering. At the same time, public ledgers expose addresses, protocol tags, and event signatures that can create strong shortcuts. A credible benchmark for trace-native learning must therefore measure shift and shortcuts directly.

The verified benchmark evidence establishes the scale and difficulty of the setting. The raw-log universe contains 1.35 billion transactions with logs and 5.01 billion raw log rows from 2021–2025. The task-labelled benchmark contains 311.97 million labelled transaction instances and 2.33 billion matched raw logs. The forward split trains on 2021–2024, validates on 2025H1, and tests on 2025H2. The test split is more compositionally complex than the training period and contains real transfer settings: Uniswap v4 and Ekubo v1 are absent from training, and 26.2 million DEX test transactions occur on pool or liquidity-infrastructure identifiers absent from training.

The benchmark also shows why shortcut diagnostics belong in the main evidence. Before masking, generic event topics are highly predictive. After excluding common token contracts and generic Transfer, Approval, Swap, and Sync topics, DEX top-topic concentration falls sharply, while liquidation detection remains more concentrated. This motivates the benchmark's task structure: DEX classification is the main large-scale

representation-learning task, and liquidation detection is a rare-event stress test that should be interpreted with stronger masking controls.

Finally, the paper defines MEM-BL as a reference modeling protocol and specifies the prediction artifacts required for performance claims. The verified trace-feature baseline suite shows that shallow execution-complexity features are strong, and the verified linear and neural topic/emitter suites show that Ethereum-native event signatures and emitter identities are also strong, especially for liquidation detection. Model-superiority claims must demonstrate gains beyond both trace-count and topic/emitter baselines, especially on unseen-topic and unseen topic–emitter slices, and under shortcut-controlled conditions. This choice follows the benchmark-paper principle that auditable data construction, leakage control, evaluation design, and reproducibility are part of the contribution. The resulting benchmark can compare parser, richer feature-based, sequence, graph, generic Transformer, and trace-native models under a shared, shortcut-aware protocol.

Ethics statement

The study uses public blockchain data and does not involve human participants, animal subjects, private off-chain user data, or attempts to deanonymize wallet owners.

Funding

The research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability

The study uses public Ethereum blockchain data. The current verified ETH-TraceBench package supports the benchmark-construction claims through a warehouse query package, Q01–Q13 benchmark manifests, split summaries, label manifests, and shortcut-diagnostic outputs. Additional model-performance claims require experiment configurations, saved per-seed predictions, table-generation code, environment files, and smoke-test subsets. Full raw blockchain mirrors may need to be reconstructed from public chain data rather than redistributed. The added lightweight neural baseline uses the same fixed topic/emitter feature file as the linear symbolic baselines and reports metrics from saved per-seed prediction files.

Declaration of competing interest

Declarations of interest: none.

Acknowledgements

Omitted for double-blind review.

References

Koh, P.W., Sagawa, S., Marklund, H., Xie, S.M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R.L., Gao, I., et al., 2021. WILDS: A benchmark of in-the-wild distribution shifts. In: Proceedings of the 38th International Conference on Machine Learning, PMLR 139, pp. 5637–5664.

Stockman, S., Lawson, D., Werner, M., 2026. EarthquakeNPP: A benchmark for earthquake forecasting with neural point processes. Transactions on Machine Learning Research.

Hickey, L., Harrigan, M., 2022. The Bisq decentralised exchange: On the privacy cost of participation. Blockchain: Research and Applications 3, 100029. doi:10.1016/j.bcra.2021.100029.

Laurent, A., Brotcorne, L., Fortz, B., 2022. Transaction fees optimization in the Ethereum blockchain. Blockchain: Research and Applications 3, 100074. doi:10.1016/j.bcra.2022.100074.

Ashizawa, N., Yanai, N., Cruz, J.P., Okamura, S., 2022. Eth2Vec: Learning contract-wide code representations for vulnerability detection on Ethereum smart contracts. Blockchain: Research and Applications 3, 100101. doi:10.1016/j.bcra.2022.100101.

Chen, H., Luo, X., Shi, L., Cao, Y., Zhang, Y., 2023. Security challenges and defense approaches for blockchain-based services from a full-stack architecture perspective. Blockchain: Research and Applications 4, 100135. doi:10.1016/j.bcra.2023.100135.

Buterin, V., Illum, J., Nadler, M., Schar, F., Soleimani, A., 2024. Blockchain privacy and regulatory compliance: Towards a practical equilibrium. Blockchain: Research and Applications 5, 100176. doi:10.1016/j.bcra.2023.100176.

Huang, S., Hao, X., Sun, Y., Wu, C., Li, H., Ren, W., Choo, K.K.R., 2024. TMAS: A transaction misbehavior analysis scheme for blockchain. Blockchain: Research and Applications 5, 100197. doi:10.1016/j.bcra.2024.100197.

Gan, J., Su, J., Lin, K., Zheng, Z., 2025. FinanceFuzz: Fuzzing smart contracts with financial properties. Blockchain: Research and Applications 6, 100301. doi:10.1016/j.bcra.2025.100301.

Arceri, V., Merenda, S.M., Negrini, L., Olivieri, L., Zaffanella, E., 2025. EVMLiSA: Sound static control-flow graph construction for EVM bytecode. Blockchain: Research and Applications, 100384. doi:10.1016/j.bcra.2025.100384.

Wang, D., Zhang, S., Tong, N., Yan, X., Li, F., 2025. A smart contract vulnerability detection method based on comprehensive code graph and temporal feature enhancement networks. Blockchain: Research and Applications, 100416. doi:10.1016/j.bcra.2025.100416.

Alnajjar, M.I., Kiraz, M.S., Kardas, S., Al-Bayatti, A., 2026. Formalization of MEV: Modeling, security requirements, and mitigation strategies. Blockchain: Research and Applications, 100470. doi:10.1016/j.bcra.2026.100470.

Schar, F., 2021. Decentralized finance: On blockchain- and smart contract-based financial markets. Federal Reserve Bank of St. Louis Review 103, 153-174. doi:10.20955/r.103.153-74.

Werner, S.M., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., Knottenbelt, W.J., 2022. SoK: Decentralized finance (DeFi). In: ACM Conference on Advances in Financial Technologies, pp. 30-46.

Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A., 2020. Flash Boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. In: IEEE Symposium on Security and Privacy, pp. 910-927. doi:10.1109/SP40000.2020.00040.

Qin, K., Zhou, L., Afonin, Y., Lazzaretti, L., Gervais, A., 2021. Attacking the DeFi ecosystem with flash loans for fun and profit. In: Financial Cryptography and Data Security, pp. 3-32.

Qin, K., Zhou, L., Gervais, A., 2022. Quantifying blockchain extractable value: How dark is the forest? In: IEEE Symposium on Security and Privacy.

Zhou, L., Qin, K., Cully, A., Livshits, B., Gervais, A., 2021a. On the just-in-time discovery of profit-generating transactions in DeFi protocols. In: IEEE Symposium on Security and Privacy.

Zhou, L., Qin, K., Torres, C.F., Le, D.V., Gervais, A., 2021b. High-frequency trading on decentralized on-chain exchanges. In: IEEE Symposium on Security and Privacy, pp. 428-445. doi:10.1109/SP40001.2021.00041.

Lehar, A., Parlour, C.A., 2025. Decentralized exchange: The Uniswap automated market maker. Journal of Finance 80, 321-374. doi:10.1111/jofi.13405.

Angeris, G., Chitra, T., 2020. Improved price oracles: Constant function market makers. arXiv:2003.10001.

Angeris, G., Agrawal, A., Evans, A., Chitra, T., Boyd, S., 2021. Constant function market makers: Multi-asset trades via convex optimization. arXiv:2107.12484.

Milionis, J., Moallemi, C.C., Roughgarden, T., Zhang, A.L., 2022. Automated market making and loss-versus-rebalancing. arXiv:2208.06046.

Park, A., 2023. The conceptual flaws of decentralized automated market making. Management Science 69, 6731-6751. doi:10.1287/mnsc.2021.02802.

Weintraub, B., Torres, C.F., Nita-Rotaru, C., State, R., 2022. A Flash(bot) in the pan: Measuring maximal extractable value in private pools. arXiv:2206.04185.

Alipanahloo, Z., Hafid, A.S., Zhang, K., 2024. Maximal extractable value mitigation approaches in Ethereum and Layer-2 chains: A comprehensive survey. arXiv:2407.19572.

Materwala, H., Naik, S.M., Taha, A., Abed, T.A., Svetinovic, D., 2024. Maximal extractable value in decentralized finance: Taxonomy, detection, and mitigation. arXiv:2411.03327.

Zhang, M., Zhang, X., Zhang, Y., Lin, Z., 2020. TXSPECTOR: Uncovering attacks in Ethereum from transactions. In: USENIX Security Symposium, pp. 2775-2792.

Wu, S., Wang, D., He, J., Zhou, Y., Wu, L., Yuan, X., He, Q., Ren, K., 2024. DeFiRanger: Detecting DeFi price manipulation attacks. IEEE Transactions on Dependable and Secure Computing 21, 1633-1648. doi:10.1109/TDSC.2023.3346888.

Chen, Z., Liu, Y., Beillahi, S.M., Li, Y., Long, F., 2024. Demystifying invariant effectiveness for securing smart contracts. arXiv:2404.14580.

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S., 2013. A fistful of Bitcoins: Characterizing payments among men with no names. In: Proceedings of the Internet Measurement Conference, pp. 127-140.

Ron, D., Shamir, A., 2013. Quantitative analysis of the full Bitcoin transaction graph. arXiv:1207.7139.

Kalodner, H., Goldfeder, S., Chator, A., Moeser, M., Narayanan, A., 2017. BlockSci: Design and applications of a blockchain analysis platform. arXiv:1709.02489.

Weber, M., Domeniconi, G., Chen, J., Weidele, D.K.I., Bellei, C., Robinson, T., Leiserson, C.E., 2019. Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. arXiv:1908.02591.

Bellei, C., Xu, M., Phillips, R., Robinson, T., Weber, M., Kaler, T., Leiserson, C.E., Arvind, Chen, J., 2024. The shape of money laundering: Subgraph representation learning on the blockchain with the Elliptic2 dataset. arXiv:2404.19109.

Elmougy, Y., Liu, L., 2023. Demystifying fraudulent transactions and illicit nodes in the Bitcoin network for financial forensics. arXiv:2306.06108.

Cheng, L., Zhu, F., Wang, Y., Liang, R., Liu, H., 2024. From asset flow to status, action, and intention discovery: Early malice detection in cryptocurrency. ACM Transactions on Knowledge Discovery from Data 18, Article 50. doi:10.1145/3626102.

Duan, C., Lei, H., Shi, J., Wang, L., 2026. DIBA: Effective transaction topology-based detection of illicit Bitcoin addresses. ACM Transactions on Knowledge Discovery from Data 20, Article 58. doi:10.1145/3800947.

Khodabandehlou, S., Hashemi Golpayegani, A., 2024. FiFrauD: Unsupervised financial fraud detection in dynamic graph streams. ACM Transactions on Knowledge Discovery from Data 18, Article 111. doi:10.1145/3641857.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998-6008.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171-4186. doi:10.18653/v1/N19-1423.

He, K., Chen, X., Xie, S., Li, Y., Dollar, P., Girshick, R., 2022. Masked autoencoders are scalable vision learners. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16000-16009. doi:10.1109/CVPR52688.2022.01553.

Du, M., Li, F., Zheng, G., Srikumar, V., 2017. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298. doi:10.1145/3133956.3134015.

Guo, H., Yuan, S., Wu, X., 2021. LogBERT: Log anomaly detection via BERT. In: International Joint Conference on Neural Networks, pp. 1-8. doi:10.1109/IJCNN52387.2021.9534113.

Hamilton, W.L., Ying, R., Leskovec, J., 2017. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024-1034.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2018. Graph attention networks. In: International Conference on Learning Representations.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.M., 2020. Temporal graph networks for deep learning on dynamic graphs. arXiv:2006.10637.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K., 2020. Inductive representation learning on temporal graphs. In: International Conference on Learning Representations.

Hooi, B., Shin, K., Song, H.A., Beutel, A., Shah, N., Faloutsos, C., 2017. Graph-based fraud detection in the face of camouflage. ACM Transactions on Knowledge Discovery from Data 11, Article 44. doi:10.1145/3056563.

Tipirneni, S., Reddy, C.K., 2022. Self-supervised Transformer for sparse and irregularly sampled multivariate clinical time-series. ACM Transactions on Knowledge Discovery from Data 16, Article 105. doi:10.1145/3516367.

A Supplementary Material and Reproducibility Appendix

Appendix A reports the supplementary materials associated with the extraction, labeling, training, evaluation, auditing, and release workflow. It documents the model configuration, hyperparameter search space, baseline controls, weak-label rules, audit protocol, split manifests, shortcut tests, objective-level ablations, representation diagnostics, runtime scaling, optional external validation, error-analysis templates, responsible-release safeguards, and reproducibility package structure.

A.1 Detailed artifact and release-interface tables

The following tables support the main-text discussion of prediction artifacts, package components, runtime reporting, loader requirements, and developer reporting rules. They are kept in the appendix so that the main paper emphasizes benchmark construction, diagnostics, and verified baseline findings.

A.2 Hyperparameter grids and selected configurations

Table 33 reports the MEM-BL base configuration proposed for the reference rerun. Table 34 reports the pretraining grid and the selected values to use when regenerating model results. Table 35 reports the downstream fine-tuning grid. The values define an execution protocol and should be linked to saved checkpoints and prediction artifacts before numerical performance claims are reported.

Table 26: Verification status of performance-related claims

Claim type	Required artifact	Artifact status
Numeric trace-feature baselines	Per-seed prediction files for DEX and liquidation under the forward split	Included for Dummy-Majority, Dummy-Stratified, TraceStats-LR, TraceStats-RF, and TraceStats-GB
Ethereum-native topic/emitter baselines	Per-seed prediction files using topic, emitter, topic–emitter, and topic/emitter+trace views	Included for TopicBag-SGD, EmitterBag-SGD, TopicEmitterHash-SGD, and TopicEmitterTrace-SGD
Lightweight neural topic/emitter baseline	Per-seed prediction files using hashed topic, emitter, and topic–emitter tokens	Included for TopicEmitterHashMLP across three seeds; interpreted as a neural symbolic baseline rather than MEM-BL superiority
MEM-BL model superiority	MEM-BL checkpoints and per-seed predictions evaluated under the same splits as the baseline suite	Not reported numerically in this paper; no superiority claim made
Protocol-transfer performance	Predictions restricted to test transactions from project/protocol combinations absent from training	Not reported numerically in this paper; required before a model-superiority claim
Pool or liquidity-infrastructure holdout performance	Predictions restricted to DEX test transactions whose pool or liquidity-infrastructure identifier is absent from training	Not reported numerically in this paper; required before a model-superiority claim
Address-masked and topic-masked robustness	Predictions under contract-masked, topic-masked, and address+topic-masked inputs	Not reported numerically in this paper; required before a model-superiority claim
Label-efficiency performance	Per-seed predictions for 1%, 5%, and higher label fractions using fixed subsampling seeds	Not reported numerically in this paper; required before a model-superiority claim
Objective ablations and runtime scaling	Per-ablation predictions, configuration files, runtime logs, hardware metadata, and seed logs	Not reported numerically in this paper; required before a model-superiority claim

A.3 Baseline fairness tables

Table 36 reports the baseline fairness controls used in the evaluation. Baselines receive the same temporal splits, label manifests, early-stopping rule, validation metric, optimizer family when applicable, and seed list. The main difference is the input structure each baseline is designed to consume.

Table 36: Baseline fairness controls

Baseline	Input information	Parameters	Fairness control
Rule parser	Protocol rules, decoded events, token-flow reconciliation	Not applicable	Uses only rules defined before test period
Random forest	Hand-engineered event counts, token-flow features, gas bins	2.1M split nodes	Same train/validation/test manifests
XGBoost	Hand-engineered features plus protocol tags	4,000 trees	Same feature cutoff and temporal split
DeepLog-style model	Ordered event topics and role tags	41M	Same sequence length cap

Baseline	Input information	Parameters	Fairness control
LogBERT-style model	Event-topic tokens and local context	128M	Same pretraining budget as MEM-BL
Supervised Transformer	Event, asset, contract, depth, gas embeddings	151M	No unlabeled pretraining
Generic masked Transformer	Same tokenized trace as MEM-BL but generic MLM objective	151M	Same hidden size and pretraining steps
Address-contract GNN	Address, contract, token-transfer graph	96M	Same temporal graph cutoff
Temporal Graph Network	Dynamic address-contract graph with timestamped transfers	118M	Same neighbor sampling budget
TGAT-style model	Time-encoded transfer graph	126M	Same validation metric and seeds
Contrastive transaction model	Positive and negative transaction-window pairs	149M	Same hard-negative source pool
MEM-BL	Event topics, typed attributes, call depth, order, flow, block context	164M	Full framework

A.4 Operational extraction scripts

The release repository contains scripts named according to their execution order. The list below shows the extraction workflow. The names are intentionally operational so that each manuscript table can be traced back to an executable step.

```
00_config.yaml
01_extract_blocks_transactions.sql
02_extract_receipts_logs.sql
03_extract_internal_calls.py
04_decode_abis_with_time_cutoffs.py
05_resolve_proxy_implementations.py
06_extract_token_transfers.sql
07_join_price_oracle_snapshots.py
08_construct_block_local_windows.py
09_build_contract_bytecode_families.py
10_generate_candidate_labels.py
11_reconcile_token_flows.py
12_apply_ambiguity_flags.py
13_build_split_manifests.py
14_export_pretraining_shards.py
15_export_supervised_benchmark.py
16_audit_sample_builder.py
17_table_generation_from_predictions.py
```

A.5 Verified label-source rules

The verified supervised benchmark in this revision uses two high-precision label sources. DEX labels are drawn from `crosschain.dex.trades`; liquidation labels are drawn from `ethereum.lending.liquidations`. These sources define task-labelled transaction instances for evaluation. They are not described as a complete taxonomy of all Ethereum economic behavior.

A.6 Manual audit template

Manual audit is reserved for claims that rely on human verification or ambiguous transaction interpretation. The verified benchmark-construction diagnostics do not require a manual audit table, but any model-performance extension

Table 27: Prediction-file schema required for final TMLR performance tables

Field group	Required columns	Purpose
Run identifiers	`run_id`, `model`, `seed`, `config_hash`	Links each row to a reproducible model configuration and random seed.
Task and split	`task`, `label`, `split`, `condition`	Separates DEX, liquidation, temporal, protocol-transfer, holdout, and masking evaluations.
Instance identifiers	`transaction_hash` or sanitized transaction key	Allows paired tests, deduplication, and reproducibility checks without changing metric values.
Ground truth and predictions	`y_true`, `y_pred`, `score` or class-probability columns	Supports macro-F1, rare recall, AUPRC, false-positive rate, and calibration metrics.
Holdout metadata	`project`, `protocol`, `pool_id`, `protocol_holdout`, `pool_holdout`	Recreates protocol-transfer and pool-holdout tables from the same predictions.
Masking metadata	`address_masked`, `topic_masked`, `generic_topic_excluded`	Verifies that shortcut-controlled results correspond to the intended input condition.

Table 28: Model-comparison tables to regenerate from prediction artifacts

Table	Primary grouping	Metrics
Primary downstream results	Task × model × seed	Macro-F1, accuracy, rare recall, AUPRC or false-positive rate
Protocol-transfer results	Task × holdout status × model	Macro-F1, degradation from full test, confidence interval
Pool-holdout results	DEX pool-holdout status × model	Macro-F1, rare recall, degradation from seen-pool test
Shortcut-controlled results	Masking condition × model	Macro-F1, degradation from unmasked input, paired bootstrap comparison
Label-efficiency results	Label fraction × model	Macro-F1, AUPRC, standard error across fixed subsampling seeds
Ablation and runtime results	Objective or model variant × seed	Macro-F1, throughput, memory, training time

should audit high-confidence false positives, high-confidence false negatives, protocol-transfer errors, and pool-holdout errors. Auditors should observe only information available at or before the focal block.

1. Load the transaction key, block number, receipt status, decoded logs, raw topics, internal calls where available, token transfers, gas features, and adjacent block-local context when it is part of the model input.
2. Mark all protocol metadata that was observable before the focal block. Ignore labels, registry entries, or ABI information first observed after the focal block.
3. For DEX labels, verify whether the transaction participates in decentralized-exchange trade activity and whether route complexity creates ambiguity.
4. For liquidation labels, verify repayment source, collateral movement, sale path, and any flash-loan leg when these fields are observable.
5. Mark ambiguity when evidence supports more than one label or when a required trace component is missing.
6. Record the exact event indices and transfer indices that support the assigned audit category.

A.7 Split manifests

Table 40 reports the split and diagnostic manifests. The table records the logical manifest names produced by the warehouse query package; release packages should additionally attach file hashes for the exact archived artifacts.

Table 29: ETH-TraceBench package components and release status

Component	Status in this version	Benchmark role
Construction queries	Verified	Recreate Q01–Q13 raw-universe counts, task labels, split tables, protocol holdouts, pool or liquidity-infrastructure holdouts, topic diagnostics, and raw-log completeness checks.
Monthly chunk outputs	Verified	Avoid one-shot warehouse joins and provide auditable intermediate artifacts for large-scale counts and split summaries.
Split and holdout manifests	Verified	Fix train, validation, test, protocol-transfer, and pool-holdout subsets so new models are evaluated on the same instances.
Shortcut-diagnostic manifests	Verified	Record address concentration, topic concentration, and generic-topic-excluded diagnostics used to define masking and robustness conditions.
Prediction schema validator	Included	Checks that new model outputs contain task, split, condition, model, seed, label, score, and holdout metadata before metrics are computed.
Metric-regeneration script	Included	Recomputes the verified trace-feature and topic/emitter baseline tables and defines the required pipeline for performance tables.
Topic/emitter extraction scripts	Included and executed	Extract ordered topic, emitter, and topic–emitter symbolic views from raw logs for domain-native baseline training.
Topic/emitter trainer scripts	Included and executed	Train TopicBag-SGD, EmitterBag-SGD, TopicEmitterHash-SGD, and TopicEmitterTrace-SGD prediction archives; metrics verified from prediction files.
Standardized data loader	Specified for leaderboard release	Should expose the same splits, labels, masks, and holdout flags through a stable API for downstream method development.
Baseline prediction archive	Included/verified for numeric trace-feature and topic/emitter suites; specified for leaderboard release for richer model families	Required before reporting model performance, ablations, masking results, holdout results, or runtime beyond the current verified baseline tables.
Leaderboard manifest	Specified for leaderboard release	Should define accepted metrics, tuning rules, required seeds, artifact submission format, and public/private test policies.

Table 30: Runtime and computational-cost artifacts for release reporting

Pipeline stage	Required log fields	Evidence in this version	Additional release requirement
Warehouse extraction	Query ID, SQL hash, date range, row count, runtime, and API/warehouse limits	Query chunks and manifests verify scale and counts	Add wall-clock/runtime logs for all full extraction chunks
Feature construction	Input chunk hash, output feature hash, row count, and dropped smoke/test rows	Clean 911,291-row feature sample and manifest included	Add deterministic feature-builder hash and schema version
Baseline training	Model, seed, hyperparameters, hardware, runtime, memory, and software versions	Prediction files for five trace-count baselines, four linear topic/emitter baselines, and one neural topic/emitter baseline across three seeds verified	Add training runtime and memory logs for all reported models
Metric regeneration	Prediction hash, metric script hash, grouping variables, and table output hash	Verified metric tables regenerated from 60 prediction files	Release exact verifier script and output hashes with the leaderboard package
Leaderboard submission	Model-card, prediction files, configuration, and declared tuning protocol	Interface specified for leaderboard submission	Add public/private split policy and artifact validation checklist

Table 31: Target loader and evaluation interface for a complete ETH-TraceBench release

Interface element	Exposed object	Purpose
Split loader	`load_split`	Returns the fixed train, validation, or test instance set without date leakage.
Condition loader	`mask_condition`	Applies address, topic, address+topic, or generic-topic-excluded inputs consistently.
Holdout selector	`holdout_flags`	Recreates protocol-transfer and pool-holdout evaluation subsets from verified manifests.
Trace representation	`event_fields`	Provides event-stream inputs without requiring each method to rejoin raw logs.
Prediction writer	`prediction_file`	Saves transaction-level predictions in the schema required by Table 27.
Metric evaluator	`metric_script`	Computes macro-F1, AUROC/AUPRC, rare recall, calibration, and shift degradation from saved outputs.
Audit report	`audit_report`	Confirms split integrity, masking condition, row counts, and holdout membership before results are reported.

Table 32: Minimum reporting checklist for ETH-TraceBench submissions

Checklist item	Required reporting
Split integrity	Confirmation that training uses only 2021–2024 labels and validation uses only 2025H1.
Core metrics	Macro-F1, AUROC/AUPRC where applicable, rare-class recall, calibration, runtime, and memory.
Shift reporting	Separate scores for full temporal test, protocol transfer, and pool/liquidity-infrastructure holdout.
Shortcut controls	Separate address-masked, topic-masked, address+topic-masked, and generic-topic-excluded evaluations.
Seeds and uncertainty	At least three fixed seeds, with confidence intervals or paired bootstrap tests over the same test instances.
Prediction artifacts	Transaction-level or sanitized-key predictions with model, seed, task, split, condition, score, and holdout metadata.

Table 33: Selected MEM-BL base configuration

Component	Selected value
Event embedding dimension	256
Transaction-local Transformer layers	6
Block-context Transformer layers	2
Attention heads	8
Feed-forward dimension	1024
Dropout	0.10
Attention dropout	0.10
Maximum events per focal transaction	256
Block-local context window	8 preceding and 8 following transactions
Address-contract memory horizon	50,000 blocks
Continuous feature projection	32-dimensional linear projection
Optimizer	AdamW
Weight decay	0.01
Gradient clipping	1.0
Mixed precision	Enabled
Random seeds	11, 17, 23, 31, 43

Table 34: Pretraining grid and selected values

Parameter	Candidate values	Selected value
Learning rate	$\{1e{-}4, 5e{-}5, 2e{-}5\}$	$5e{-}5$
Batch size	$\{256, 512, 1024\}$	512
Warmup steps	$\{5k, 10k, 20k\}$	10k
Maximum steps	$\{150k, 250k, 350k\}$	250k
Topic masking rate	$\{0.10, 0.15, 0.20\}$	0.15
Attribute masking rate	$\{0.10, 0.20, 0.30\}$	0.20
Span removal rate	$\{0.05, 0.10, 0.15\}$	0.10
Order corruption rate	$\{0.05, 0.10, 0.15\}$	0.10
Flow corruption rate	$\{0.05, 0.10, 0.20\}$	0.10
Context temperature τ	$\{0.03, 0.07, 0.10\}$	0.07
Loss weights $(\lambda_a, \lambda_c, \lambda_o, \lambda_f)$	$(1, 1, 0.5, 1)$; $(1, 0.5, 0.5, 1)$; $(1, 1, 1, 1)$	$(1, 1, 0.5, 1)$
Hard negatives per span	$\{8, 16, 32\}$	16

Table 35: Fine-tuning grid and selected values

Parameter	Candidate values	Selected value
Learning rate	$\{3e{-}5, 1e{-}5, 5e{-}6\}$	$1e{-}5$
Task-head learning rate	$\{1e{-}4, 5e{-}5, 2e{-}5\}$	$5e{-}5$
Batch size	$\{64, 128, 256\}$	128
Epochs	$\{5, 10, 15\}$	10
Early stopping patience	$\{2, 3, 5\}$	3
Class weighting	None; inverse frequency; square-root inverse frequency	square-root inverse frequency
Calibration	None; temperature scaling	temperature scaling
Frozen-layer variants	frozen, partial, full	task dependent
Validation metric	macro-F1; AUPRC; rare recall	macro-F1

Table 37: Extraction checks and exclusion rules

Check	Failure condition	Treatment
Receipt-status consistency	Receipt status contradicts emitted-log presence	Exclude from supervised benchmark
Log-index uniqueness	Duplicate transaction hash and log index	Retain only if mirror checksum agrees
Trace completeness	Internal call tree missing focal call depth	Exclude from trace tasks
Token-decimal availability	Token decimals unavailable before focal block	Flag amount features as unknown
Price snapshot availability	No price or oracle state before focal transaction	Use missingness flag, no future imputation
Proxy mapping cutoff	Implementation relation only known after focal block	Do not apply mapping
Bridge direction	Bridge event has inconsistent source and destination token flow	Flag ambiguous

Table 38: Verified label-source rules

Task label	Source table	Transaction key	Interpretation
DEX swap/trade	`crosschain.dex.trades`	`transaction_hash`	High-precision positive label for decentralized-exchange trade activity.
Liquidation	`ethereum.lending.liquidations`	`transaction_hash`	High-precision positive label for lending-liquidation activity.

Table 39: Audit sampling template for model-error analysis

Audit stratum	Sample	Double-coded	Purpose
Full DEX test errors	1,000	1.00	Check common false-positive and false-negative patterns.
DEX protocol-transfer errors	1,000	1.00	Check failures on protocols absent from training.
DEX pool-holdout errors	1,000	1.00	Check whether failures concentrate on new pool identifiers.
Liquidation test errors	1,000	1.00	Check rare-event false negatives and collateral-flow ambiguity.
Address+topic-masked errors	1,000	1.00	Check whether shortcut-controlled failures differ from full-input failures.

Table 40: Revised split and diagnostic manifest summary

Manifest	Description	Rows or scale
Q01 raw corpus manifest	Monthly raw-log transaction and event counts, 2021–2025	60 months
Q02 event-complexity manifest	Monthly counts by 1–2, 3–10, 11–50, and 51+ log bins	240 rows
Q03 DEX protocol manifest	DEX project/protocol coverage and volume, 2021–2025	60 chunks
Q04 liquidation manifest	Lending-liquidation transaction and project coverage	60 chunks
Q05 weak-label manifest	DEX and liquidation weak-label source counts	120 rows
Q06 temporal label manifest	Train, validation, and test label counts	6 rows
Q08 protocol-holdout manifest	Project/protocol presence across train, validation, and test	69 rows
Q09 pool-holdout manifest	Pool or liquidity-infrastructure identifiers by holdout status	491,997 identifiers
Q10–Q11 shortcut manifests	Event-topic concentration before and after generic-topic exclusions	Top-20 topic tables
Q12 final split summary	Final task, label, split, transaction, raw-log, topic, and emitter counts	6 rows
Q13 zero-log diagnostic	Raw-log join completeness in the 2025H2 test split	2 rows

A.8 Shortcut diagnostics and withheld performance tables

The verified shortcut diagnostics in this version are dataset diagnostics rather than model-performance results. Q10 shows that the ERC-20 `Transfer` topic appears in most labelled 2025H2 DEX and liquidation transactions after common-token contract exclusion. Q11 shows that, after removing common infrastructure contracts and generic event topics, DEX topic concentration falls sharply: the top remaining topic appears in 14.35% of DEX-labelled test transactions, while the remaining traces span 6,783 topics and 97,084 emitting contracts. Liquidation remains more concentrated: the top remaining topic appears in 53.53% of liquidation-labelled test transactions. These diagnostics justify topic-masked and address+topic-masked evaluation but do not themselves prove model robustness.

A.9 Objective ablations and representation diagnostics to regenerate

Objective ablations, representation diagnostics, and runtime-scaling curves require saved per-seed predictions, configuration hashes, and runtime logs. They are not reported numerically beyond the trace-feature and topic/emitter baseline suites. This appendix records the required outputs and prevents unverifiable values from being inserted into the manuscript.

A.10 Optional external MEV matching manifest

The verified benchmark contains DEX and liquidation labels. It does not include a verified external MEV prediction artifact. If an observable-MEV auxiliary validation set is added later, the release package should include the source, retrieval date, block range, transaction hash or sanitized key, transaction index, pool address when available, original

Table 41: Verified shortcut diagnostics available before model reruns

Diagnostic	DEX result	Liquidation result	Interpretation
Top topic share before generic-topic exclusion	80.7%	95.2%	Generic event signatures create strong shortcut risk.
Top topic share after generic-topic exclusion	14.35%	53.53%	DEX retains diverse residual event structure; liquidation remains concentrated.
Unique topics after generic-topic exclusion	6,783	434	DEX is the stronger large-scale representation-learning benchmark.
Unique emitting contracts after generic-topic exclusion	97,084	1,797	Liquidation is better framed as a rare-event stress test.
Zero raw-log share in 2025H2	0.035%	0.000%	Raw-log join incompleteness is negligible for the test split.

Table 42: Ablation and diagnostic artifacts required before numerical reporting

Output family	Required files	Required checks
Objective ablations	Per-seed predictions for full MEM-BL and each removed objective	Same split, same label manifests, same checkpoint-selection rule.
Masking robustness	Predictions under address-masked, topic-masked, and address+topic-masked inputs	Confirm that masked fields are unavailable to the model at inference and not reintroduced through metadata.
Representation diagnostics	Embeddings for the contract-masked test set and metadata for task, protocol, and contract identity	Report economic-role clustering and contract-identity clustering from the same embedding file.
Runtime scaling	Hardware metadata, event-window length, throughput, memory, and model configuration logs	Report throughput and memory without mixing hardware or batch-size settings.

label, mapped label, match status, and exclusion reason. No numerical MEV validation table is reported in this version because the supporting matching manifest and saved predictions are not supplied.

A.11 Error-analysis template

Manual error analysis is part of the artifact contract for any new model run. The error-analysis file should record the task, split, model, seed, transaction key, true label, predicted label, confidence score, protocol, holdout status, masking condition, and analyst-coded error category. Transaction hashes can be delayed, sanitized, or replaced by stable salted keys in the public manuscript package.

A.12 Responsible-release checklist

Table 44 records the responsible-release safeguards attached to the benchmark package. The checklist is included because public-chain data can support defensive monitoring while also creating deanonymization, exploit-discovery, or strategic-trading risks if released without use constraints.

Table 43: Error-analysis categories to populate after verified predictions

Category	Trace pattern to inspect	Why it matters
Aggregator routing ambiguity	Multi-hop routes with repeated transfers and pool interactions	Benign routing can resemble arbitrage or rare mechanisms.
Liquidation callback complexity	Borrow, repay, collateral transfer, and sale occur through nested callbacks	Call depth can obscure repayment source and collateral-sale path.
Generic-topic shortcut	Predictions remain confident after Transfer, Approval, Swap, or Sync topics are removed	Indicates possible dependence on event signatures rather than compositional structure.
New-protocol failure	Errors concentrate in Uniswap v4, Ekubo v1, or other project/protocol combinations absent from training	Indicates weak protocol-transfer performance.
Pool-holdout failure	Errors concentrate in pool or liquidity-infrastructure identifiers absent from training	Indicates contract memorization or poor local generalization.
Zero-log or incomplete-trace case	Labelled transaction has missing or incomplete raw-log join	Separates data-quality failures from modeling failures.

Table 44: Responsible-release checklist

Checklist item	Status
Do not publish mappings from wallet addresses to real-world identities	Satisfied
Delay or sanitize examples involving active vulnerabilities	Satisfied
Release model outputs for defensive monitoring, not exploit automation	Satisfied
Separate public-chain identifiers from human-attribution claims	Satisfied
Document known false-positive sources for any auxiliary labels	Required for auxiliary labels
Include ambiguity flags for uncertain transaction windows	Satisfied
Exclude transactions requiring non-public off-chain information	Satisfied
Provide intended-use and excluded-use language in the repository	Satisfied

A.13 Reproducibility checklist

Table 45: Reproducibility checklist

Item	Artifact	Status
Data extraction scripts	Q01–Q13 query and runner manifests	Verified
Label construction scripts	DEX and liquidation label-source queries	Verified
Split manifests	Temporal, protocol-transfer, and pool-holdout manifests	Verified
Configuration files	YAML files for all model and baseline runs	Included for baselines
Random seeds	Fixed seed list for all supervised and ablation runs	Verified for baselines

Item	Artifact	Status
Saved predictions	Per-seed prediction CSV files	Verified for reported baselines
Table-generation scripts	Recreate all model-performance tables from saved predictions	Added
Audit instructions	Annotator guide, adjudication form, ambiguity definitions	Template added
External validation matching	Source map and exclusion log for any auxiliary validation	Optional
Responsible-release notes	Wallet-identity, exploit, and vulnerability safeguards	Required
Smoke-test subset	Small public reproducibility subset	Included for extraction scripts
Environment file	Python package versions and execution notes	Included for baseline scripts

A.14 Repository structure for the release package

```
eth-tracebench-release/
  README.md
  configs/
    pretrain_membl_base.yaml
    finetune_economic.yaml
    finetune_liquidation.yaml
    baselines/*.yaml
  extraction/
    01_extract_blocks_transactions.sql
    02_extract_receipts_logs.sql
    03_extract_internal_calls.py
    04_decode_abis_with_time_cutoffs.py
    05_resolve_proxy_implementations.py
  labels/
    weak_rules.yaml
    flow_reconciliation.py
    ambiguity_flags.yaml
    audit_protocol.md
  manifests/
    temporal_split.jsonl
    protocol_transfer_split.jsonl
    contract_holdout_split.jsonl
    external_mev_matching.jsonl
  models/
    membl_encoder.py
    objectives.py
    task_heads.py
  evaluation/
    metrics.py
    shortcut_tests.py
    representation_diagnostics.py
    table_generation.py
  release/
    responsible_release_checklist.md
    data_availability_statement.md
```