The Enemy of Reproducibility is Opacity: What's Inside the Elliptic Bitcoin Dataset (and Why It Is Wrong)

Miroslav Šafář[a,*], Jan Pluskal[a,*], Vladimír Veselý[a,*] and Ondřej Ryšavý[a]

[a]*Faculty of Information Technology, Brno University of Technology, Božetěchova 2, Brno 612 00, , Czech Republic*

ARTICLE INFO

Keywords:
digital forensics
blockchain analysis
blockchain forensics
Bitcoin analysis
transaction classification
Elliptic dataset
reverse engineering
neural networks
graph neural networks
Elliptic dataset feature meaning
Elliptic dataset feature semantics
Elliptic dataset feature description
ML data leakage

Abstract

Machine-learning methods for Bitcoin transaction classification are frequently evaluated on the Elliptic Bitcoin Dataset (2019), and many recent "state-of-the-art" results rely on this benchmark. Yet Elliptic's feature construction process was not fully disclosed, making the benchmark difficult to reproduce and limiting operational use: without clear feature semantics, practitioners cannot compute the required inputs for previously trained models on new transactions encountered in real investigations. Beyond hindering its real-world application, this opacity has also obscured what information the dataset actually contains and how it propagates across standard experimental splits. In this paper, we reverse engineer the Elliptic dataset to recover the meaning of almost all features and to reconstruct key aspects of the dataset's derivation. This analysis enables a systematic audit of evaluation practices and leads to an unsettling finding: the dataset's commonly used training and testing splits are not properly isolated. We provide evidence of this leakage and discuss why it can compromise the validity and comparability of Elliptic-based evaluations. Finally, we propose a solutions to the problems that occured during Elliptic dataset construction, accompanied with a methodology for building Bitcoin transaction classification datasets, with explicit feature semantics, leakage-resistant splitting that respects temporal and graph dependencies, and full pipeline transparency. We release an open-source tool that collects and transforms raw Bitcoin blockchain data into a transparent transaction dataset, enabling the development of models suitable for real-world forensic applications.

1. Introduction

Over the past fifteen years, blockchain-based systems have been exploited for money laundering, terrorist financing, fraud, dark-market commerce, and, more recently, sanctions evasion. As a result, identifying illicit activity has become a key task for law enforcement, while anti–money laundering (AML) compliance is increasingly required for financial institutions that support cryptocurrency transactions. Consequently, forensic analysis of blockchain data, including transaction and address analysis, wallet clustering, entity attribution, and fund tracing across the public ledger (often supplemented with off-chain intelligence), has become an important part of digital forensics.

In recent years, numerous machine-learning approaches have been proposed for Bitcoin transaction classification, reporting promising performance. These methods provide a basis for automated, near–real-time assessment of Bitcoin transactions, reducing reliance on expert judgement and manually crafted heuristics. However, many such approaches (e.g., Alarab and Prakoonwit (2022), Lo, Kulatilleke, Sarhan, Layeghy and Portmann (2023), Chai, You, Yang, Pu, Xu, Cai and Jiang (2022), Duan, Zheng, Gao, Wang, Feng and Wang (2024), Li and He (2023)) have been evaluated on the Elliptic Bitcoin dataset[1], released by Weber, Domeniconi, Chen, Weidele, Bellei, Robinson and Leiserson (2019). Although the dataset is large enough for both traditional machine-learning methods (e.g, Linear Regression, Random Forest) and neural network-based approaches (e.g., Multi-Layer Perceptron, Graph Neural Networks), it does not fully disclose how features are constructed. This lack of transparency makes the models trained on this dataset useless in real-world applications – without clear feature semantics, practitioners cannot compute the required inputs for models trained on Elliptic when analysing new transactions in real investigations. In this paper, we therefore analyse this popular dataset and reverse engineer its feature construction to provide a foundation for real-world forensic applications of these models.

During our analysis, we discovered an unsettling finding, that Elliptic Bitcoin dataset suffers from data leakage problem, i.e., information leakage between commonly used training and test splits, which, according to Kaufman, Rosset, Perlich and Stitelman (2012) and Kapoor and Narayanan (2023), can result in an overestimation of model performance. We present evidence of this data leakage and propose a reproducible, forensically grounded methodology for building Bitcoin transaction classification datasets, with explicit feature semantics and a splitting strategy that respects temporal and graph dependencies while avoiding test and validation leakage. Finally, we present an open-source tool designed for dataset construction following the proposed methodology, enabling the creation of a transparent Bitcoin transactions dataset and the subsequent use of trained models in real-world forensic applications.

isafar@fit.vut.cz (M. Šafář); pluskal@fit.vut.cz (J. Pluskal); veselyv@fit.vut.cz (V. Veselý); rysavy@fit.vut.cz (O. Ryšavý)

https://www.fit.vut.cz/person/isafar/ (M. Šafář); https://www.fit.vut.cz/person/pluskal/ (J. Pluskal); https://www.fit.vut.cz/person/veselyv/ (V. Veselý); https://www.fit.vut.cz/person/rysavy/ (O. Ryšavý)

ORCID(s): 0009-0006-2409-316X (M. Šafář); 0000-0002-8304-3684 (J. Pluskal); 0000-0002-6346-2152 (V. Veselý); 0000-0001-9652-6418 (O. Ryšavý)

[1] https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

1.1. Contributions

The main contributions of this article are:

- description of the semantics of more than 90 % of the Elliptic Bitcoin dataset features,
- identification of all transactions included in Elliptic Bitcoin dataset,
- identification of a fundamental issue in the Elliptic Bitcoin dataset, the information leakage between commonly used training and test splits, that can lead to the overestimation of the performance of the evaluated machine-learning methods,
- proposed solutions to identified issues of the Elliptic dataset,
- a blueprint for constructing Bitcoin transaction datasets with transparent feature semantics and real-world forensic applicability,
- an open-source tool for constructing a Bitcoin transaction dataset from blockchain data and transaction labels, such as those provided with the original Elliptic Bitcoin dataset.

1.2. Paper structure

This paper is structured as follows. Section 2 provides a brief overview of the state of the art in machine-learning-based blockchain forensics and shows that many state-of-the-art methods rely on the Elliptic dataset as a benchmark. Section 3 then describes the Elliptic dataset, our reverse-engineering process, and the recovered transaction feature semantics. Section 4 presents the identified data leakage mechanisms and their consequences for the validity of evaluations that use the Elliptic dataset as a benchmark. Section 5 describes our mitigation strategies and introduces tooling that can be used to construct a dataset consistent with these mitigation strategies. Section 6 explains why we cannot publish the resulting dataset and discusses the licensing limitations of Elliptic. Finally, Section 7 concludes the paper.

2. Related work

Lee, Maharjan, Ko and Hong (2020) applied a multilayer perceptron (MLP) and Random Forest to binary transaction classification, labelling as illicit those associated with Silk Road. Li, Cai, Tian, Xue and Zheng (2020) similarly focused on transaction characteristics, using an autoencoder built from an MLP and an LSTM to capture temporal information for classifying addresses as benign or linked to illicit activity (primarily ransomware and extortion). In both works, classification relied on features derived from local transaction/address properties rather than graph structure.

To address this limitation, Weber et al. (2019) introduced the Elliptic Bitcoin dataset and evaluated conventional machine-learning methods and graph convolutional networks (GCNs). Random Forest achieved the best performance on Elliptic, but only when augmented with expert-engineered neighbourhood features, suggesting that GNNs may be able to replace manual feature design. Subsequent work incorporated temporal dynamics into GNNs on Elliptic, including EvolveGCN (Pareja, Domeniconi, Chen, Ma, Suzumura, Kanezashi, Kaler, Schardl and Leiserson (2020)) and temporal-GCN (Alarab and Prakoonwit (2022)), both combining graph convolutions with recurrent units (LSTM/GRU), with Alarab and Prakoonwit (2022) additionally employing active learning.

Chai et al. (2022) claimed that several studies argue that standard GNNs behave as low-pass filters and may therefore underperform on anomalous nodes, which is relevant when illicit activity manifests as structural or feature outliers. To address this issue, Chai et al. (2022) proposed AMNet to capture and adaptively combine low- and high-frequency signals via attention. Duan et al. (2024) introduced DGA-GNN, which aggregates information separately from node groups stratified by predicted legitimacy and then fuses it using attention. Wang, Tsai and Du (2024) proposed RM-GANets, a reinforcement-learning-based framework with attention, reporting state-of-the-art results as of late 2024, although the underlying dataset and code were not made available for verification. Lo et al. (2023) used self-supervised learning (Deep Graph Infomax with a GIN encoder) to enrich node representations and then classified them with Random Forest, outperforming previous methods.

The extension of the Elliptic Bitcoin dataset, Elliptic++[2], was published by Elmougy and Liu (2023). Assuming that Elliptic's transaction features do not include address-level information[3] the authors augmented the transaction feature vectors with address-level features derived from the addresses participating in each transaction. In addition, they introduced a derived dataset of blockchain actors (wallet addresses), referred to as the Elliptic++ actor dataset.

Beyond transaction-level classification, Song, Zhang, Zhang, Park, Gu and Yu (2025) proposed SGNN (Social Attention Graph Neural Network), motivated by the argument that address-level classification can be preferable for anti–money laundering tasks and leveraging the Elliptic++ actor dataset. The transaction classification pipeline therefore consists of two stages. First, labels are predicted for all addresses participating in a transaction. These predicted address-level labels are then used to construct transaction-level features, which are subsequently employed to classify the transactions themselves. SGNN currently reports the strongest performance for transaction classification.

Finally, recent work expands the scope to entity-graph subgraph classification: Bellei, Xu, Phillips, Robinson, Weber, Kaler, Leiserson, Arvind and Chen (2024) introduced Elliptic2[4], a dataset that contains 49 million nodes representing address clusters and 196 million edges representing transactions. Bellei et al. (2024) evaluated subgraph classifiers on the Elliptic2 dataset and GLASS (Wang and Zhang (2022)) classifier performed the best.

[2] https://github.com/git-disl/EllipticPlusPlus

[3] In Section 3.2, we present a fact that the transaction features do include address-level information.

[4] https://www.kaggle.com/datasets/ellipticco/elliptic2-data-set

3. Elliptic Dataset

In this section, we describe the Elliptic Bitcoin dataset, outline the reverse-engineering process, and present the semantics of all recovered features.

The Elliptic Bitcoin dataset, introduced by Weber et al. (2019) for anti–money laundering (AML) research, contains 203,769 Bitcoin transactions and 234,355 directed payments flows (edges). For comparison, the Bitcoin blockchain comprises more than 1.3 billion confirmed transactions as of January 2026. A total number of 46,564 Eliptic bitcoin dataset transactions are labelled as licit or illicit. The distinction is based on their association with entities belonging to licit categories (e.g., exchanges, wallet providers, miners) versus illicit ones (e.g., scams, malware, terrorist organisations, ransomware, Ponzi schemes), the rest of the transactions are unlabeled. According to the authors, transaction labels are derived from the initiating entity (i.e., the entity controlling the input addresses, or an address cluster). However, the released dataset does not include the underlying mapping to real-world entities used to assign these labels, so the label assignments cannot be independently verified.

This dataset is a transaction graph where nodes represent transactions and directed edges represent funds flowing between these transactions. The graph itself consists of 49 separate subgraphs. Each subgraph represents a time step, uniformly distributed over an interval of approximately two weeks. The time step can be thought of as an instantaneous "snapshot" of the transactions that appeared on the blockchain at a specific time – each time step is a single connected component of transactions that appeared on the blockchain within less than three hours of each other. The time steps span the period from 2016-01-01 to 2017-10-02. Consequently, only the last four time steps were collected after the activation of Segregated Witness[5] (SegWit), and none were collected after the activation of Taproot[6].

In the context of a machine learning inductive task (such as the classification of previously unseen transactions), Weber et al. (2019) employed a 70:30 temporal split between training and test data for their evaluation benchmark. The first 34 time steps were used for training, while the remaining 15 time steps constituted the test set. This split has subsequently been adopted in later studies to enable comparable results. Accordingly, we assume the same dataset split in our analysis.

3.1. Feature semantics

Each node has associated 166 features. According to Weber et al. (2019), the first 94 features represent local information about the transaction – number of inputs/outputs, transaction fee, output volume and aggregated figures such as average BTC received (spent) by the inputs/outputs and average number of incoming (outgoing) transactions associated with the inputs/outputs. The remaining 72 features, called aggregated features, are obtained by aggregating transaction information one-hop backward/forward (maximum, minimum, standard deviation and correlation coefficients of the neighbour transactions). The semantics of each dataset feature were not published at the time to protect the intellectual property of the Elliptic company. In addition, all transaction features were anonymised to hide their original values. This significantly limits interpretability, forensic applicability, and independent validation of experimental results. To assess the suitability of the dataset for real-world forensic use and to understand potential sources of data leakage, we performed a systematic reverse engineering of the dataset's structure and transaction features.

3.2. Reverse Engineering

Initial transaction mapping. A critical prerequisite for reverse engineering was the ability to map Elliptic transaction identifiers to real Bitcoin transactions. We leveraged a publicly available mapping[7] covering approximately 99.5 % of Elliptic transactions, which links dataset-specific transaction ids to their corresponding Bitcoin transaction hashes. For identification of these transactions, they successfully localised the features for transaction volume, fee, number of inputs/outputs and number of unique addreses on inputs/outputs. Combined with knowledge of data collection time frame and graph structure, they were able to identify those 99.5 % of transactions. In this paper, we identify the rest. At first, we retrieved the full transaction data for each identified transaction using the Blockbook API[8].

Feature Anonymization. Exploratory analysis revealed that, with the exception of the explicit time-step feature, all numerical features exhibit a zero mean and unit variance. This strongly indicates that Elliptic features were anonymized using z-score normalization. Under this assumption, *differences between anonymized values preserve relative differences in the original feature space* up to a constant scaling factor, enabling the partial reconstruction of original values and distributions.

The key step in reconstructing original feature values is therefore the estimation of this scaling factor. To achieve this, we employed two strategies based on the observed feature value distributions.

Feature distributions. During exploratory analysis, two dominant distributional patterns were observed (see Figure 1):

1. *Lower-bounded, right-skewed distributions.* In the context of blockchain transactions, this pattern corresponds to non-negative integer quantities such as transaction value, transaction fee, number of inputs, number of outputs, or their aggregate statistics (e.g., minimum, maximum, and mean). These quantities are naturally lower-bounded by zero, and the probability of larger values typically decreases as the magnitude increases. Under this assumption, we estimated the scaling factor by mapping a unit increment in the

[5] https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[6] https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki

[7] https://www.kaggle.com/datasets/alexbenzik/deanonymized-995-pct-of-elliptic-transactions

[8] https://github.com/trezor/blockbook

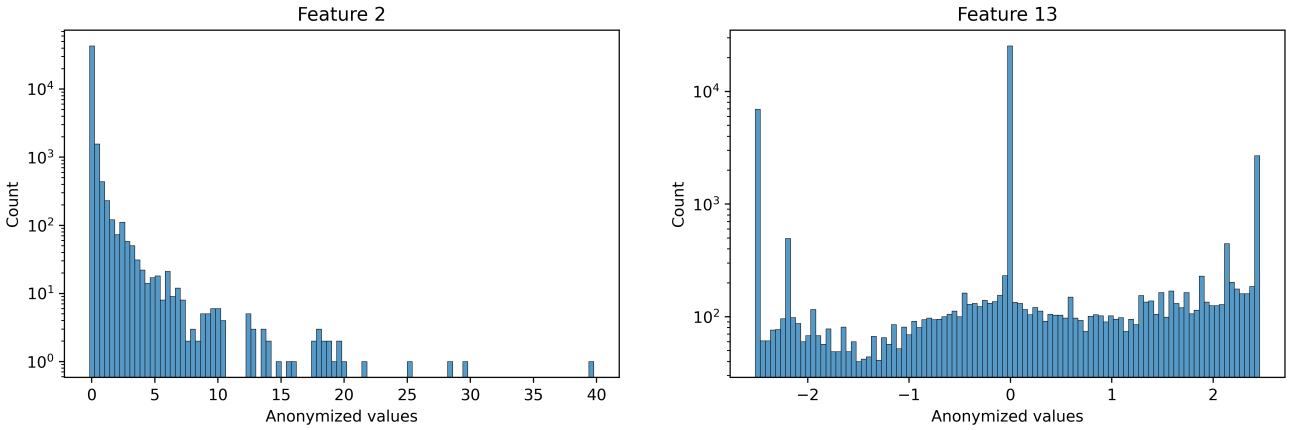

Figure 1: Distributions of Elliptic transaction features 2 and 13. Feature 2 exhibits a lower-bounded, right-skewed distribution, whereas feature 13 exhibits a bounded, tri-modal distribution. In our exploratory analysis, we observed that most features follow one of these two distributional patterns.

original feature space to either the minimum or the modal difference between distinct anonymized values.

2. *Bounded, tri-modal distributions.* This pattern likely corresponds to features whose values lie within bounded ranges such as $[0, 1]$ or $[-1, 1]$. Based on the Elliptic dataset description indicating that aggregated features include correlation coefficients, which are bounded in $[-1, 1]$, we estimated the scaling factor by mapping the anonymized feature range to $[-1, 1]$, thereby projecting anonymized values back into that interval.

After rescaling anonymized feature values using the estimated scaling factor, we grouped transactions according to low, intermediate, and high feature values. For each group, we compared transaction data obtained from Blockbook and searched for features exhibiting consistent values within groups and marginally different values across groups. Candidate feature interpretations were then validated by comparing reconstructed values against the rescaled anonymized features.

Subsequently, we identified a consistent construction pattern for aggregated transaction features. All aggregated features were computed using the minimum, maximum, standard deviation, mean, and Pearson and Spearman correlation coefficients. The correlation coefficients capture correlations between aggregated values and their positional indices within the aggregated sequences.

Local transaction features. Despite its name, we found out that only the first 21 of the local transaction features are scoped directly to the transaction itself, while the remaining features within this group correspond to aggregated global features of Bitcoin addresses participating in the transaction. Transaction-scoped features include transaction volume, transaction fee, number of inputs and outputs, number of unique input/output addresses, and aggregated statistics of input and output values. We were unable to identify the semantics of the remaining three transaction-scoped features.

Address-scoped features include the total BTC received and sent, available balance, number of incoming transactions, number of outgoing transactions, and address lifetime (defined as the time between the first and last observed transaction). These features are aggregated separately for input and output addresses. Crucially, the reference point for all address-scoped features corresponds to the time of dataset construction, specifically block at height 575,059.

Aggregated neighborhood features. Aggregated neighborhood features follow a similar construction process to address-scoped features. Transaction volume, block height, transaction fee, number of inputs, and number of outputs are aggregated separately for predecessor and successor transactions. Notably, these features are not computed solely from neighborhood transactions included in the released dataset, but from all neighboring transactions available at the time of Elliptic dataset construction.

Transaction identification. Using the recovered feature semantics, we were able to identify all transactions included in the Elliptic dataset. We started from the publicly available mapping covering approximately 99.5 % of Elliptic transactions and used these identified transactions to determine the approximate time window (and corresponding block range) for each time step. For each time step, we fetched all blocks that could plausibly contain transactions from that window, iterated over all transactions in those blocks, and computed the deanonymized transaction features together with the aggregated address-level features evaluated at block height 575,059. We then compared these computed features against the denormalized values of the deanonymized features for the remaining (previously unidentified) Elliptic transactions. This procedure yielded a unique match for each Elliptic transaction, except in cases where multiple transactions shared identical feature vectors; for these collisions, we assigned Elliptic ids according to their original ordering.

Validation. The reconstructed features were validated through an end-to-end reconstruction of the Elliptic dataset.

Using the identified transactions, we recomputed the deanonymized features from blockchain data using the recovered semantics, applied z-score normalization consistent with the original preprocessing, and compared the resulting feature vectors transaction by transaction against the Elliptic dataset. This resulted in exact agreement for all deanonymized features, providing strong evidence that the recovered semantics are correct.

Summary. We recovered the semantics of 91 of the 94 local features and 60 of the 72 aggregated neighborhood features. The exact feature-semantic mapping is available on GitHub[9].

4. Problems of the Elliptic Dataset

In the previous section, we described the semantics of the transaction features contained in the Elliptic dataset and showed that feature anonymization is performed via z-score normalization. In this section, we analyze data leakage issues present in the Elliptic dataset. Each identified leakage instance is categorized using the taxonomy proposed by Kapoor and Narayanan (2023). We focus on leakage mechanisms that can lead to overly optimistic evaluation results and compromise the validity of experimental conclusions.

4.1. Pre-processing on training and test set

The first leakage issue stems from the dataset's anonymization procedure. Feature values are z-score normalized, however, the normalization parameters (mean and standard deviation) were computed using the entire dataset rather than the training split alone. Because z-score normalization is typically part of the pre-processing pipeline, this introduces information from the test distribution into the training process. Put differently, the training pipeline is allowed to depend on statistics estimated from the test set, which violates the requirement that test data remain unseen during training. In the taxonomy of Kapoor and Narayanan (2023), this falls under *Lack of clean separation between training and test sets*, specifically *Pre-processing on training and test sets*.

4.2. Temporal leakage

The second leakage issue is less obvious but more consequential. Unlike the normalization leakage, it does not arise from an implementation error during dataset construction; rather, it is inherent to the semantics of the provided features. As described in the previous section, the local feature vector of a transaction combines (i) attributes derived from the transaction itself and (ii) aggregates derived from address-level features. Crucially, these address-level features are computed once at a single reference time t_f, which occurs after the end of the test period (see Figure 2). As a result, feature vectors for training (and validation) transactions implicitly encode address activity that occurs in the future relative to those transactions, including activity from transactions in the test split. This constitutes look-ahead (temporal) leakage.

In the taxonomy of Kapoor and Narayanan (2023), this corresponds to *Test set is not drawn from the distribution of scientific interest*, specifically the subcategory *Temporal leakage*. The key requirement is that the training pipeline must not incorporate information that would only become available after the prediction time.

In our setting, because address aggregates are defined using information available at t_f, the effective feature construction time for the training split is t_f. Since t_f lies after the test period, the training features incorporate information from a time that is later than the test examples on which they are evaluated, violating the intended causal ordering and inflating performance by allowing the model to leverage information that would not be available at deployment.

Temporal leakage can also be characterized formally using the legitimacy framework of Kaufman et al. (2012). In this framework, each feature value x is associated with an observability time t_x, and each target element y is associated with a prediction time t_y. A feature is considered legitimate for predicting y if and only if

$$t_x < t_y \iff x \in \text{legit}(y). \tag{1}$$

A model contains leakage with respect to a target element y if it uses at least one observational input $x \notin \text{legit}(y)$.

In the Elliptic dataset, we treat transaction features as observational inputs and the transaction label as the prediction target. Consider a transaction y from the test split, for which a prediction is required a few moments after the transaction's occurrence time at t_y. While transaction-intrinsic features satisfy $t_x < t_y$, the aggregated address-level features violate this condition. These aggregates are computed at a single reference time t_f, which occurs after the end of the test period (see Figure 2). Therefore, for these features we have $t_x = t_f > t_y$. By the contrapositive of Equation 1, this implies $x \notin \text{legit}(y)$.

As a consequence, any model trained or evaluated using these address-level aggregates violates the no-time-machine requirement of Kaufman et al. (2012) and contains temporal leakage. The model is effectively trained using information that becomes observable only after the prediction time, including information derived from transactions in the test split itself.

One might argue that temporal leakage can be avoided by redefining the prediction time to the global feature construction time t_f, thereby rendering all address-level aggregates legitimate. However, this redefinition changes the task from predicting labels for newly observed transactions (as in deployment) to retrospectively labeling historical transactions with access to information that becomes available only at t_f. As such, it no longer reflects the original operational or scientific objective of transaction-level illicit activity detection. Moreover, this task redefinition does not resolve the issue of nonindependence between training and test samples described in Section 4.3.

[9] https://github.com/nesfit/DeanonymizedEllipticBitcoinDataset

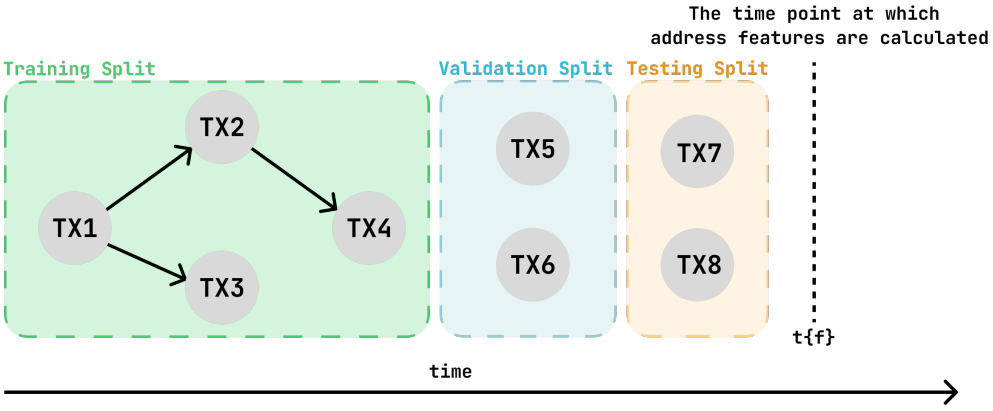

Figure 2: Illustration of a temporal data leakage instance in the Elliptic dataset. Transactions are split chronologically into training, validation, and test sets, but the address-level features used to construct transaction local features are computed once at a single reference time t_f (after the end of the test period) and then reused for all transactions involving the same address. Consequently, the feature vectors of training (and validation) transactions implicitly reflect address activity that occurs in the future, including activity from transactions in the test split, yielding look-ahead leakage and overoptimistic evaluation. In the taxonomy of Kapoor and Narayanan (2023), this corresponds to the case where the *Test set is not drawn from the distribution of scientific interest*, specifically the subcase of *Temporal leakage*, since the model is effectively trained using information "from the future" that would not be available at prediction time.

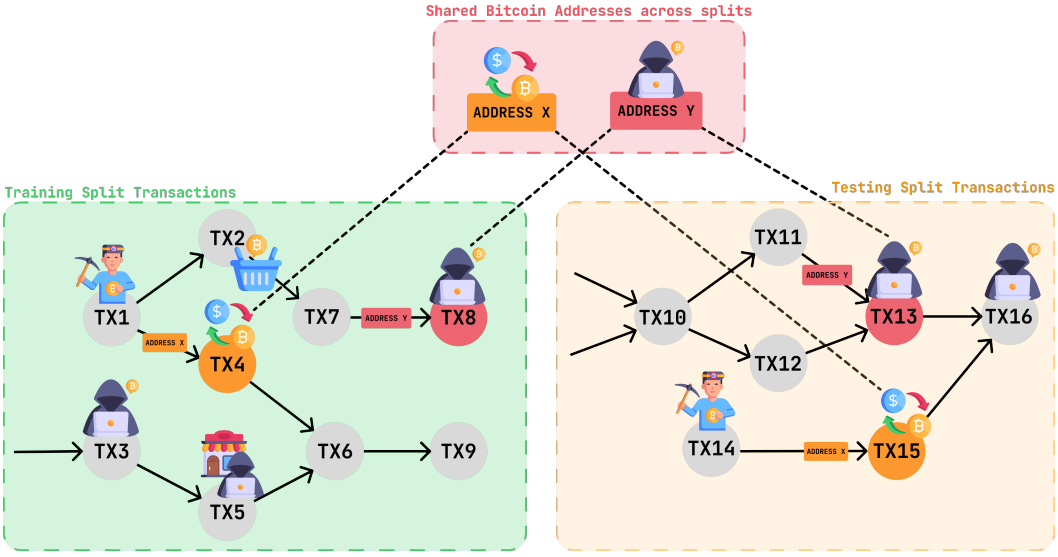

Figure 3: Illustration of a data leakage instance in the Elliptic dataset. The local features of a transaction include aggregated features of its input/output Bitcoin addresses. These address-level features are computed once at a single reference time and are therefore identical for all transactions that interact with the same address. Because addresses appear across the training, validation, and test splits, models can exploit address-specific signals that were already observed during training, constituting entity-level (address) leakage. In the taxonomy of Kapoor and Narayanan (2023), this corresponds to the case where the *Test set is not drawn from the distribution of scientific interest*, specifically the subcase of *Nonindependence between training and test samples*. Instead of evaluating whether a model can classify transactions associated with previously unseen malicious actors (the intended task), the test set partially evaluates transactions linked to actors (addresses) already present in the training data.

4.3. Nonindependence between training and test samples

In addition to temporal leakage, the Elliptic dataset exhibits a second major issue: nonindependence between training and test samples. In the taxonomy of Kapoor and Narayanan (2023), this issue also falls under *Test set is not drawn from the distribution of scientific interest*.

The source of this nonindependence lies in the construction of transaction-level features. Local transaction features include aggregated statistics of the input and output Bitcoin addresses involved in a transaction. These address-level aggregates are identical for all transactions that interact with the same address and are reused unchanged across time. Because addresses appear in multiple temporal snapshots and are shared across the training, validation, and test splits

(see Figure 3), transactions in the test set are not statistically independent from those in the training set.

As a consequence, models trained on the Elliptic dataset can exploit address-specific signals observed during training when making predictions on test transactions that involve the same addresses. This allows the model to partially memorize entity-level behavior rather than infer illicit activity from transaction-level patterns alone. The resulting evaluation therefore overestimates generalization performance, as it does not measure the model's ability to classify transactions associated with previously unseen entities (the intended task), but it measures the classification of transactions linked to entities (addresses) already present in the training data.

The impact of this leakage is further amplified by the dataset's labeling procedure. As described in the original Elliptic dataset paper (Weber et al. (2019)), transaction labels are inferred from the labels of input addresses. Consequently, if an address is labeled as illicit, all transactions that spend outputs from that address inherit the same label. When such addresses occur across training and test splits, both features and targets become coupled through shared entities, further increasing the dependence between samples. As a result, test-set performance is driven not only by shared address-level features but also by consistent entity-level labels, reinforcing the extent to which evaluation reflects address re-identification rather than transaction-level generalization.

4.4. Other problems blocking real-world application

Beyond the leakage mechanisms discussed above, the Elliptic dataset exhibits additional design choices that hinder real-world forensic applicability and limit its suitability for training graph neural networks, one of its primary intended use cases. In particular, the dataset is organized into discrete time steps, where each time step forms an isolated transaction graph containing only transactions that occurred within a fixed temporal window. This sampling strategy truncates transaction neighborhoods by omitting predecessor and related transactions that would normally be reachable in the full Bitcoin transaction graph. Consequently, some non-coinbase transactions appear to have no input transactions, and others have only a subset of their true inputs present (see Figure 4). This induced graph structure departs from realistic forensic deployment, where the analyst (or model) has access to the complete transaction history up to the prediction time and can evaluate a transaction in its full causal context.

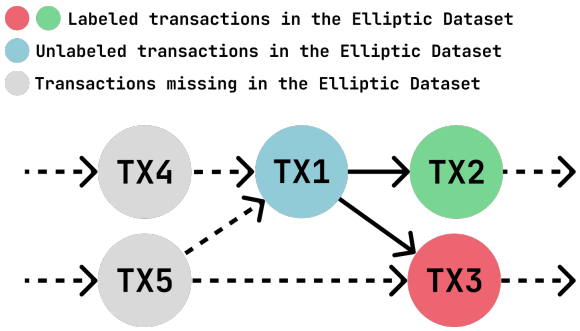

Figure 4: Illustration of the missing neighborhood problem in the Elliptic dataset. By design, each time step contains only transactions that appeared on the blockchain within the corresponding time window. As a result, the transaction graph within a single time step is an isolated component with truncated neighborhoods. This leads to situations where non-coinbase transactions appear to have no input transactions (e.g., TX1), or where only a subset of a transaction's true inputs is present (e.g., TX3). Such incomplete neighborhoods do not reflect the real-world forensic deployment setting, where the full historical blockchain up to the prediction time is available.

5. Proposed Solutions

In this section, we offer a solution to correct the mistakes made during the construction of the Elliptic dataset and enable the creation of a Bitcoin transaction dataset with transparent feature semantics and the applicability of the trained models in real-world forensic scenarios, which was not possible with the models trained on the Elliptic dataset.

Pre-processing on training and test set. Ensuring the pre-processing of the newly created dataset needs to be done only using information from the training dataset. In other worlds, for z-score normalization, the mean and standard deviation should be calculated on the training dataset and whole dataset should be processed based on there statistics.

Temporal leakage. To mitigate temporal leakage, caused by computing address-level features at a single future reference time, we propose using time-scoped address features aligned with each transaction. Specifically, address-level feature values should be computed as of the time the transaction occurred on the blockchain, or at a fixed relative delay after it (e.g., +1 hour, +1 day) to incorporate information that becomes available shortly after the transaction[10]. In real-world deployments, introducing such a delay would correspondingly delay transaction-level predictions; however, it avoids temporal leakage. As discussed in Section 4.2, temporal leakage arises when features incorporate information computed after the prediction time. When the same relative delay is applied consistently to both feature computation and prediction, the feature legitimacy rule is preserved.

Missing transaction neighborhood. To address the missing transaction neighborhood, we propose constructing each labeled transaction's neighborhood using hop-distance sampling (i.e., including all transactions within a fixed number of hops in the payment-flow graph), and releasing the hop-distance metadata as part of the dataset. We prefer hop-based sampling over alternative sampling strategies (such as time window sampling in the Elliptic dataset) because it aligns with how modern graph neural networks (e.g., Kipf and Welling (2017), Veličković, Cucurull, Casanova, Romero, Liò and Bengio (2018)) perform message passing: an L-layer GNN can only aggregate information from nodes

[10] Usability of this approach is a subject of further research. Our current implementation considers the time when transaction occured on the blockchain without adding any additional delay.

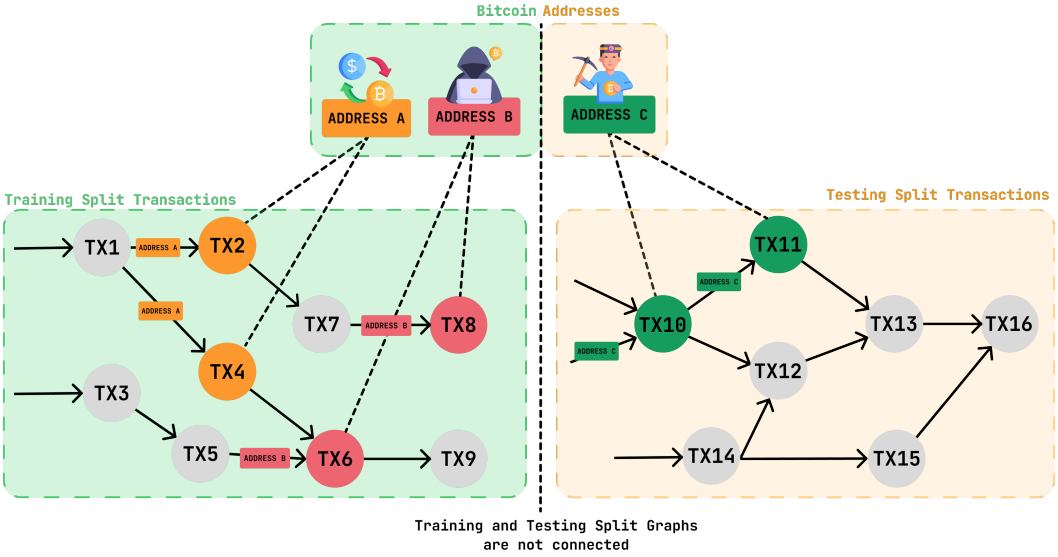

Figure 5: Illustration of a correct separation between training and test splits. Even after adding bipartite edges between transaction nodes and their participating address nodes, the split graphs remain disconnected. Equivalently, an address that participates in any transaction in the training split must not participate in any transaction in the test split, and vice versa.

within at most L hops. Therefore, the hop limit can be chosen to match the intended model depth, and the provided hop-distance information makes the effective aggregation boundary explicit to practitioners.

Nonindependence between training and test samples. This issue is the most difficult to mitigate because it does not arise solely from dataset construction choices but from the provenance of the transaction labels. We support the strategy of inferring transaction labels from address labels, since linking addresses to real-world entities is typically more tractable than labeling individual transactions directly. However, the resulting training, validation, and test graphs must satisfy a strict independence requirement. When constructing graphs from labeled transactions and their neighborhoods, the splits should be disconnected and must remain disconnected even after adding address nodes and edges representing address participation in transactions (see Figure 5). The label collection process itself is outside the scope of this paper, as it typically relies on web scraping, address clustering, and domain expertise.

5.1. Transaction features

Following the proposed solutions above and the semantics of deanonymized features of the Elliptic dataset, we propose the usage of following features:

Local features. Local transaction features can mirror the deanonymized, truly local transaction features identified in the Elliptic dataset: transaction value, transaction fee, number of inputs and outputs, and the number of unique input and output addresses. We further recommend enriching these with (i) the number of reused addresses (i.e., addresses that appear both among the inputs and outputs) and (ii) indicators of spending format and script features, such as the use of SegWit and Taproot.

Address-level features. As discussed above, address-level features should be time-scoped and aligned with each transaction. In real-world deployment, collecting such features is straightforward because computation time matches collection time, and many blockchain indexing tools (e.g., Blockbook) provide address statistics. The challenge arises during dataset construction, where address statistics must be obtained for many addresses at many different points in time. Such time-indexed address statistics are typically not available in standard indexers and therefore must be derived by processing the blockchain sequentially, block by block. An alternative is to retrieve the full transaction history for each address and compute the statistics retrospectively; however, because the number of distinct addresses in even moderately sized transaction samples can exceed the number of blocks by orders of magnitude, this approach is substantially more time- and compute-intensive.

5.2. Tooling

To support the collection of time-scoped address statistics and the construction of transaction features, we developed an open-source tool that extracts address statistics for all participating addresses and constructs the corresponding transaction-level feature vectors. The tool is available on GitHub[11] under the GPL-3 license.

As input, it takes a list of labeled transaction hashes. It recursively retrieves transaction data via the Blockbook API and expands the transaction neighborhood by following input/output links until the specified x-hop subgraph is fully populated with transaction metadata. From the resulting subgraph, the tool extracts all participating addresses together with the block heights at which they appear, and then invokes

[11] https://github.com/nesfit/BitcoinTxSubgraphBuilder

a Bitcoin blockchain parsing component to compute time-scoped address statistics.

The parser streams the blockchain block-by-block from the Bitcoin Core RPC interface, iterates over all transactions, maintains its own UTXO database, and updates an address-statistics store. Address statistics are computed with respect to a target block height: for each (address, target height) record, the statistics reflect only activity observed up to and including that height. After parsing completes, the tool constructs transaction-level features from the retrieved transaction metadata and the corresponding time-scoped address statistics. As output, it produces labeled transactions augmented with their neighborhood transactions and feature vectors. Features are not normalized by default, enabling researchers to evaluate alternative normalization strategies and improving transparency of the underlying feature values. A complete list of features and their definitions is provided in the same repository.

The resulting dataset can be used both (i) to re-evaluate existing machine-learning methods previously assessed on the Elliptic dataset, whose reported performance may be compromised by leakage, and (ii) to train models for licit/illicit transaction classification that can be deployed in real-world forensic settings, enabled by the transparent and reproducible feature construction.

6. Discussion and Limitations

Elliptic is released under the CC BY-NC-ND 4.0 (Attribution–NonCommercial–NoDerivatives 4.0 International) license, which prohibits the redistribution of modified versions of the dataset. Consequently, we cannot publish an updated or "fixed" Elliptic dataset. Instead, we release tooling that can be used to construct a new dataset from Elliptic transaction labels for non-commercial, non-distributable use. The same consideration applies to the list of identified transactions: we treat the transaction mapping as a derivative artifact of the Elliptic dataset and therefore do not redistribute it. We instead describe the acquisition process for this mapping in Section 3.2.

However, any dataset derived in this way should be used only as a training split. To enable a leakage-resistant evaluation, practitioners must construct an independent test set that shares no addresses with those appearing in the Elliptic dataset, nor with addresses occurring in the x-hop neighborhoods of Elliptic transactions, to preserve independence between training and test splits (see Section 5).

7. Conclusions

This paper shows that the Elliptic Bitcoin dataset, while widely treated as a benchmark for AML transaction classification, does not reliably support the inductive evaluation claims commonly made in the literature. By reverse engineering the feature construction, we deanonymize and reconstruct more than 90 % of the original transaction features and identify all of the included transactions. We demonstrated that a portion of the so-called local transaction features encodes address-level global statistics, a fact not made explicit in the original Elliptic description. This ambiguity appears to have misled Elmougy and Liu (2023), who enriched the dataset with features that were already present. These address-level features were computed at a single global reference time "in the future" and reused across many transactions and across the train/validation/test splits. This constitutes systematic temporal leakage and entity-level nonindependence between training and test samples, additionally combined with dataset-wide preprocessing. Consequently, reported performance can be inflated by leakage and may reflect memorization of repeated entities rather than genuine generalization to previously unseen actors and future transactions.

Beyond evaluation validity, Elliptic's anonymization and the absence of published feature semantics have constrained reproducibility and forensic usability. Without a transparent specification, researchers could neither interpret the feature values nor compute the transaction features for arbitrary transactions, preventing validation on independent testing datasets or the evaluation of the trained models on newer blockchain transactions using SegWit and Taproot functionality. More importantly, investigators cannot operationalize these models, since the required features cannot be computed for arbitrary transactions.

To address these limitations, we presented a leakage-resistant and fully transparent dataset construction pipeline, implemented as an open-source tool that derives time-scoped, interpretable features from the blockchain and preserves reproducible preprocessing. The resulting dataset can be used (i) to re-evaluate existing machine-learning methods previously assessed on the Elliptic dataset, whose reported performance may be compromised by leakage, (ii) to train models for licit/illicit transaction classification that can be deployed in real-world forensic settings, enabled by the transparent and reproducible feature construction, and (iii) to develop more complex models that address novel challenges in Bitcoin transaction classification arising from protocol upgrades such as SegWit and Taproot.

Acknowledgement

This article is funded under the ISF-2024-TF2-AG-DIGITAL project entitled *COW: Countering privacy-enhancing law enforcement challenges of Cryptocurrencies, OSINT, and Wireless leverage* with Grant agreement ID: 101202767. The authors are grateful for the support and collaboration of the entire COW consortium.

References

Alarab, I., Prakoonwit, S., 2022. Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data. Neural Processing Letters 55, 1–19. doi:10.1007/s11063-022-10904-8.

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. URL: http://arxiv.org/abs/2404.19109, doi:10.48550/arXiv.2404.19109.

Chai, Z., You, S., Yang, Y., Pu, S., Xu, J., Cai, H., Jiang, W., 2022. Can Abnormality be Detected by Graph Neural Networks?, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Vienna, Austria. pp. 1945–1951. doi:10.24963/ijcai.2022/270.

Duan, M., Zheng, T., Gao, Y., Wang, G., Feng, Z., Wang, X., 2024. DGA-GNN: Dynamic Grouping Aggregation GNN for Fraud Detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11820–11828. doi:10.1609/aaai.v38i10.29067.

Elmougy, Y., Liu, L., 2023. Demystifying Fraudulent Transactions and Illicit Nodes in the Bitcoin Network for Financial Forensics, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. pp. 3979–3990. doi:10.1145/3580305.3599803.

Kapoor, S., Narayanan, A., 2023. Leakage and the reproducibility crisis in machine-learning-based science. Patterns 4, 100804. URL: https://www.sciencedirect.com/science/article/pii/S2666389923001599, doi:https://doi.org/10.1016/j.patter.2023.100804.

Kaufman, S., Rosset, S., Perlich, C., Stitelman, O., 2012. Leakage in data mining: Formulation, detection, and avoidance. ACM Trans. Knowl. Discov. Data 6. URL: https://doi.org/10.1145/2382577.2382579, doi:10.1145/2382577.2382579.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations.

Lee, C., Maharjan, S., Ko, K., Hong, J.W.K., 2020. Toward detecting illegal transactions on bitcoin using machine-learning methods, in: Zheng, Z., Dai, H.N., Tang, M., Chen, X. (Eds.), Blockchain and Trustworthy Systems, Springer Singapore, Singapore. pp. 520–533.

Li, Y., Cai, Y., Tian, H., Xue, G., Zheng, Z., 2020. Identifying Illicit Addresses in Bitcoin Network, in: Zheng, Z., Dai, H.N., Fu, X., Chen, B. (Eds.), Blockchain and Trustworthy Systems, Springer, Singapore. pp. 99–111.

Li, Z., He, E., 2023. Graph Neural Network-Based Bitcoin Transaction Tracking Model. IEEE Access 11, 62109–62120. doi:10.1109/ACCESS.2023.3288026.

Lo, W.W., Kulatilleke, G.K., Sarhan, M., Layeghy, S., Portmann, M., 2023. Inspection-L: self-supervised GNN node embeddings for money laundering detection in bitcoin. Applied Intelligence 53, 19406–19417.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., Leiserson, C., 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs, in: Proceedings of the AAAI conference on artificial intelligence, pp. 5363–5370.

Song, J., Zhang, S., Zhang, P., Park, J., Gu, Y., Yu, G., 2025. Illicit Social Accounts? Anti-Money Laundering for Transactional Blockchains. IEEE Transactions on Information Forensics and Security 20, 391–404.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y., 2018. Graph attention networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings, Curran Associates.

Wang, Q., Tsai, W.T., Du, B., 2024. RMGANets: reinforcement learning-enhanced multi-relational attention graph-aware network for anti-money laundering detection. Complex & Intelligent Systems 11, 5. doi:10.1007/s40747-024-01615-9.

Wang, X., Zhang, M., 2022. GLASS: GNN with labeling tricks for subgraph representation learning, in: International Conference on Learning Representations.

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. URL: http://arxiv.org/abs/1908.02591, doi:10.48550/arXiv.1908.02591.