



Research



**Cite this article:** Gige AM, Buschmann Alsbirk L, Coscia M. 2026 Evaluating fraud detection algorithms in a decentralized scenario. *R. Soc. Open Sci.* **13**: 251922. <https://doi.org/10.1098/rsos.251922>

Received: 6 October 2025

Accepted: 11 March 2026

**Subject Category:**

Computer science and artificial intelligence

**Subject Areas:**

artificial intelligence

**Keywords:**

financial transaction networks, fraud detection, graph transformers, graph neural networks, graph learning, money laundering, financial fraud

**Author for correspondence:**

Michele Coscia

e-mail: [mcos@itu.dk](mailto:mcos@itu.dk)

# Evaluating fraud detection algorithms in a decentralized scenario

Ada M. Gige, Lasse Buschmann Alsbirk and Michele Coscia

IT University of Copenhagen, Copenhagen, Denmark

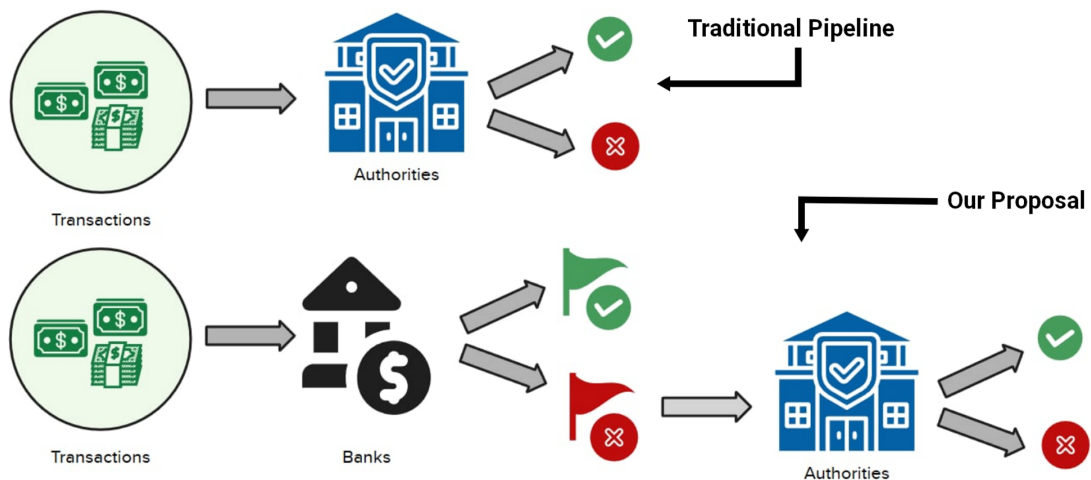
MC, 0000-0001-5984-5137

Financial fraud is an umbrella term including a vast number of illegal activities. These activities involve a significant fraction of the global economy. Traditional investigation techniques are labour-intensive and cannot scale to match the size of the issue. Machine learning has provided effective tools which deliver high accuracy in identifying transactions that could be involved in fraudulent activities. In this paper, we point out that the state-of-the-art in financial fraud detection has been applied to the unrealistic scenario of an omniscient centralized global authority which has access to all bank transactions globally. We propose a more realistic evaluation scenario, one made of two steps: first, the bank flags its own transactions using exclusively information it possesses; then only flagged transactions from all banks are analysed by the governmental authority for potential prosecution. We find that, in such a realistic scenario, the effectiveness of the state-of-the-art method for financial fraud detection decreases. Moreover, we show that in this decentralized scenario, it pays off to use simpler methods than the state-of-the-art, depending on the specific objective function the system wants to ensure.

## 1. Introduction

Financial fraud is an endemic problem in the global economy. This umbrella term includes a vast array of activities: money laundering, identity (ID) theft, investment fraud and more. The scope and size of financial fraud is massive. The United Nations estimates money laundering—only one type of fraudulent activity—to involve around 2% of the global economic activities [1]—possibly more.

When dealing with such a volume of activity, traditional investigative methods are hopeless. It is not possible to manually inspect all transactions to find patterns of suspicious activities. For



**Figure 1.** The contrast between the fraud detection pipeline in the state-of-the-art (top) against our proposal (bottom).

this reason, a number of semi-automatic approaches have been developed over the years. The classical approach is a rule-based approach, which consists of creating a simple set of criteria to flag a transaction—which is then manually investigated [2]. However, rule-based systems have a number of issues; the most important is that financial fraud is ever-evolving and cannot be consistently captured by a set of static rules, which needs to be constantly updated.

Machine learning techniques can help in this scenario, as they allow flexibility and adaptability. If the financial fraud patterns change, a machine learning system can be re-trained and updated more easily than a rule-based system. It is no surprise that we have seen in recent years the rise of accurate financial fraud detection machine learning approaches, which can deliver an improvement in accuracy when detecting illicit economic activities [3–5].

However, in our paper, we point out that we need more realistic scenarios in which to estimate the effectiveness of financial fraud detection algorithms, which we depict in figure 1. Owing to the lack of real-world data on which to evaluate methods, a common approach is to use synthetic data. The synthetic data are generated on the basis of real data, simulating every transaction for a few days in a global economy [6]. Putting aside the potential issues of the accuracy of the synthetic data itself, the common approach to evaluate financial fraud is to ingest the entire synthetic dataset—split into train and test sets—to perform the predictions (top half of figure 1).

Performing such an operation effectively means that there is one global centralized authority with perfect knowledge about the global economy: such authority can access all transactions from all banks, globally. It goes without saying that such authority does not exist in all countries, and it is not poised to exist any time soon. Even at the national level, many countries might not be able to build such infrastructure, as it would pose enormous privacy issues as well as political ones, as it could be seen as a powerful tool for surveillance. Moreover, it would put the entire burden of detecting financial fraud on central governmental authorities. Therefore, estimating the performance of a fraud detection method with the assumption that such authority exists is not a realistic use case.

With our work, we propose a different, more realistic, framework for the evaluation of fraud detection algorithms. We depict it in the bottom half of figure 1. Differently from what has been done so far, we propose to split the prediction task into two phases. In the first phase, each bank only has access to their own transactions. The bank trains its own model on its data and flags potentially fraudulent transactions to the governmental authority. In the second phase, the governmental authority collects all flagged transactions from all banks and trains its own model exclusively on the data it receives. It is the output of the governmental authority that should be evaluated, as it is this output that will ultimately decide which economic actors will be investigated for financial fraud.

The second step need not be performed automatically with machine learning techniques. For example, a report on the implementation of [7] in the United States acknowledges that the existing anti-money laundering (AML) regime ‘contemplates ageing, decades-old technology’, rather than the current AML compliance systems managed by financial institutions [7]. Without technology to assist AML workflows at agencies such as FinCEN, such processes become very labour-intensive: even if more precise, not all reports can be analysed effectively, in practice trading recall for precision.

Whether performed automatically or manually, we expect the output of a two-step pipeline to have a lower performance than the one achieved by an omniscient authority in a one-step pipeline. We base this expectation on qualitative evidence showing that pooling information from multiple banks could increase the performance of financial fraud detection [8]. The current methods in the state-of-the-art assume data pooling is in place. In our realistic scenario, we turn it off and therefore expect a degradation in performance.

This is exactly what we observe. When run using the state-of-the-art in financial fraud detection, our experiments show that there is a significant performance penalty using this framework. Using the centralized pipeline traditionally used in the state-of-the-art, we can reproduce the state-of-the-art results on the test set with F1 scores of 0.32 and 0.71—depending on the specific version of the dataset we use. However, with our two-step decentralized pipeline, the test set only reports F1 scores of 0.12 and 0.56—a significant degradation in performance. Moreover, when injecting a simpler model in one of the two steps in our framework, we can actually improve the F1 scores on the test set to 0.49 and 0.85, respectively.

In summary, the contributions of this paper are the following:

- (1) We propose a more realistic scenario for evaluating financial fraud detection algorithms;
- (2) We show that the performance of state-of-the-art financial fraud detection algorithms in this realistic scenario is lower;
- (3) We find evidence for a performance increase by *synergizing* simple and complex algorithms in realistic settings.

Our results hint at the need for a strong communication link between banks and authorities. Uncoordinated action trying to optimize each step in the financial fraud detection pipeline in isolation could lead to worse performance overall.

The data we base our results on are publicly available. The code we run is gathered from the implementation provided by the state-of-the-art,<sup>1</sup> with minimal modifications—which we provide in a separate repository [9].<sup>2</sup>

## 2. Related work

In this paper, we look at machine learning attempts to tackle the financial fraud detection problem [10,11], and we argue that we need a change in perspective in how we evaluate such attempts. We identify three classes of works that are affected differently by our arguments. The classes depend on which type of data is being analysed: cryptocurrency data, data from a single bank/institution or data from multiple banks/institutions.

In the first class, we have papers focusing on analysing cryptocurrencies. One of the most used datasets is Elliptic [12]. Examples are comparing different approaches such as deep neural network (DNN), random forest, K-nearest neighbours and naive Bayes [13]; implementing a long short-term memory (LSTM) DNN [14]; or making a guilty-by-association argument exploiting random walks on the transaction graph [15]. These works are the least affected by our argument, as they focus on a technology that is by definition decentralized and trackable. Since cryptocurrency ledgers are publicly available, it is possible to perform a train and test on the full dataset, because there are no multiple institutions handling the transactions. However, financial fraud could involve transferring funds across multiple different cryptocurrencies and traditional currencies, which will incur the issues we point out in this paper. The issues are solvable, but no paper we considered addresses them.

When focusing on a single bank/institution, the traditional approach is to use private data that is normally not shared. Examples are HAMLET implementing a transformer method [16], the use of convolutional neural networks [17], variational auto encoders [18–20] and ensemble LSTM [21], or graph neural networks (GNNs) [22–26]. Among these works, we can also find the other side of the equation, meaning using data from the governmental entity tasked with investigating financial fraud or tax evasion [27]. This highlights why such works might operate in isolation, but they are still affected by our argument. The bank and the government agency might develop methods that are optimized in isolation, but they could be suboptimal when paired with each other. Only a holistic view of the whole system can

<sup>1</sup><https://github.com/junhongmit/FraudGT>

<sup>2</sup>[https://github.com/adagige/thesis\\_public](https://github.com/adagige/thesis_public)

provide an effective way of diagnosing the efficacy of the methods, but all works in this category lack such a view.

The final category of studies normally leverages data from the IBM AML (IBM-AML) dataset [6] or similar ones [28,29]. IBM-AML enables the analysis of the economy as a whole, since it simulates data for multiple banks in different countries, including cryptocurrencies. This makes IBM-AML widely used [30,31]. A common approach—which, to the best of our knowledge, is the current state-of-the-art—is to create a GNN framework [3,32]. The GNNs are made more powerful with a number of techniques—port numbering, reverse message passing, ego ID—which allow overcoming some of the standard issues affecting the classical message-passing framework [33]. This is the most important category of work for our paper, because IBM-AML allows the holistic view enabling an accurate evaluation of the two-step bank-to-authority pipeline. However, none of the works we have identified using this dataset acknowledges this issue.

We acknowledge that IBM-AML is used with different frameworks, e.g. by creating global risk scores that are then fed to a recurrent neural network (RNN) [4]. While one could create global risk scores with public information and side-step our argument, in practical terms, only global risk scores using information handled by the banks are effective—and they are the ones used by the RNN framework. In this scenario, one would still need to have data access from multiple banks and would therefore be affected by our argument.

The other alternative is to use federated learning, a machine learning framework where all banks would share the parameters of a classifier model chosen by a central authority, effectively sharing the training results without sharing any of their sensitive data [34,35]. Federated learning has been applied to money laundering prediction tasks, with good results [36,37]. However, federated learning increases the performance of the classification task on the banks' side, but one problem persists: the central authority still needs to perform its own investigation on the flagged transactions. If there are too many flagged transactions to be checked manually, a second model still needs to be run on the law-enforcement side, meaning that federated learning alleviates but does not eliminate the issue we raise.

## 3. Framework

### 3.1. Problem statement

In the most general version of the financial fraud detection problem, we start by having a set of financial transactions  $T$ . Each transaction  $t \in T$  records a financial interaction between two accounts—one account sending money to another—and can have several attributes. Focusing on the minimal set of attributes we need to have for our argument to be applicable, each transaction  $t$  must have:

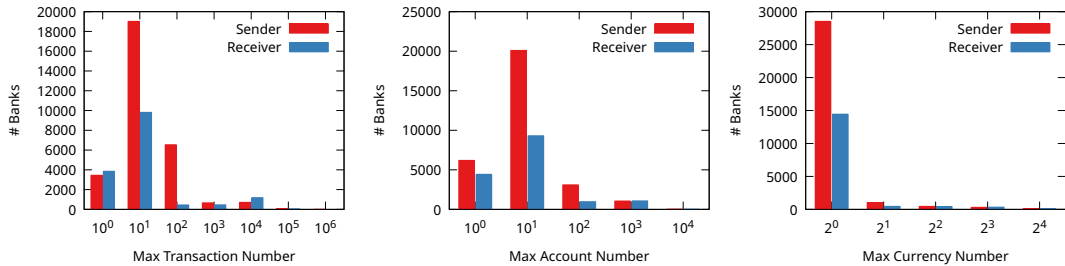
- The ID of the bank in which the account sending the money is registered;
- The ID of the bank in which the account receiving the money is registered;
- A label  $s \rightarrow \{0, 1\}$  recording the ground truth.

The label  $s$  tells us whether the transaction is suspected to be part of a financial fraud scheme—in which case  $s = 1$ , or not, and therefore  $s = 0$ .

The financial fraud detection problem is then defined as the construction of a function  $f$  that, given as input any set of transactions  $T$ , maps every transaction  $t$  on to a probability that  $s_t = 1$ . The objective is to be as accurate as possible. The evaluation criteria we focus on in this paper are precision, recall and F1 measure—which is the harmonic mean of precision and recall. Precision is the share of the accurate predictions over all the predictions made, while recall is the share of accurate predictions over all possible accurate predictions. If  $TP$  is true positives,  $FP$  is false positives and  $FN$  is false negatives, then:  $Precision = TP/(TP + FP)$ ,  $Recall = TP/(TP + FN)$  and  $F1 = 2(Precision \times Recall)/(Precision + Recall)$ .

While we present results using all of these measures, we should point out that financial fraud data have heavily unbalanced classes—i.e. the number of transactions involved in financial fraud is much lower than the ones not involved. Under the assumption that we want to capture as many fraudulent transactions as possible and we put less attention on to false positives, it makes most sense to focus on the recall of the  $s = 1$  class, which is our primary evaluation measure.

Any method addressing the financial fraud detection problem uses a number of additional attributes attached to transactions in  $T$  to build the most accurate  $f$  possible. Given that our argument is particularly relevant for works using the IBM-AML dataset, we assume without loss of generality that transactions in  $T$  also contain the attributes that can be found in that dataset, namely: timestamp of the transaction,



**Figure 2.** The number of banks ( $y$ -axis) with a given size ( $x$ -axis, bins report the upper bound of the bin) according to different measures. From left to right: number of transactions, number of accounts and number of currencies. Red for statistics referring to the bank originating the transaction, blue for the bank receiving the transaction.

IDs of origin/destination accounts within the bank, transaction amount, currency and type (wire, check, etc).

As a reference for the class imbalance issue, the IBM-AML dataset comes in two flavours, with either a high or a low amount of money laundering. In the ‘high’ money laundering incidence dataset, the fraction of  $s = 1$  transactions is approximately 0.11%, while in the ‘low’ dataset, it is approximately 0.05%.

### 3.2. Centralized

The majority of works that use the IBM-AML dataset perform a classical train-validation-test split of the data. The split is normally done *temporally*, meaning that all the data from a given set of days are used for training, then other days are selected for validation and for testing. The split is never done *by the bank*. This implicitly assumes that whatever entity is performing the financial fraud detection task has access to data from all banks.

To show why this assumption is problematic, we calculate some summary statistics on the banks in the IBM-AML dataset. The IBM-AML datasets are split into three size classes, besides the two money laundering prevalence classes we mention at the end of §3.1: large, medium and small. The difference between these classes is the number of days simulated in the dataset, which is 97, 16 and 10 days, respectively. For our point, we take summary statistics from the smallest dataset, since we want to show how this assumption is problematic also in the smallest possible case.

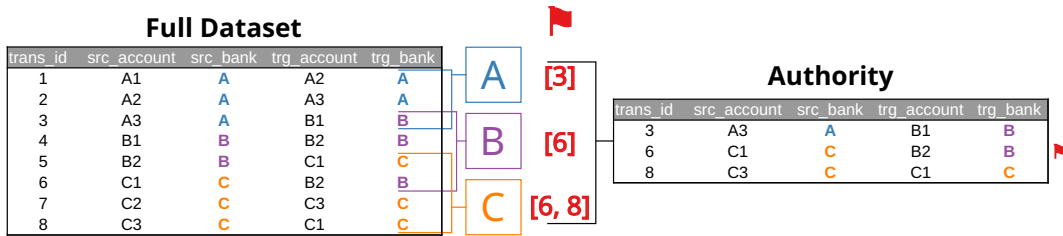
Figure 2 shows various summary statistics for the banks in the small IBM-AML dataset with high amounts of money laundering. The first observation is that the number of banks is high: in this specific dataset, we have more than 30 000. Coordinating a data infrastructure that can receive total transactions information from more than 30 000 banks is probably unfeasible.

Moreover, we can see from figure 2 that there is a large heterogeneity in bank sizes. While there are some large banks clearing hundreds of thousands of transactions from thousands of accounts in a dozen different currencies, there is also a large number of small banks with only a handful of accounts and transactions over the observation period. Such small banks might not have the manpower to send all of their transactions to a central authority and are also less likely to be able to have enough data or a suitable computing infrastructure to run sophisticated machine learning models.

### 3.3. Decentralized

In contrast with the centralized approach, we propose that a decentralized pipeline is a more realistic way of evaluating performance. Differently from the centralized pipeline, here we do not assume that there is a central authority that has access to all financial transactions. Rather, we assume that every bank has full access only to the transaction either originating or ending in an account they own. Each bank then trains its own financial fraud detection system in isolation from all other banks. The transactions flagged by such a system are then sent to the governmental authority, which pools them all together and applies its own financial fraud detection algorithm. It is the performance at the end of this two-step pipeline that matters, as it is only the governmental authority which can take legal actions against fraudsters.

Figure 3 is a more detailed reproduction of figure 1, showing how transactions are handled at each step of the pipeline. As we can see, all transactions involving accounts from different banks are used by all banks because all banks have them in their records. The flags produced by each bank are used to filter



**Figure 3.** The expanded proposed pipeline from figure 1.

the original dataset—in our example, transactions 3, 6 and 8 are the ones receiving a flag by the bank. These transactions are the only ones sent to the central authority which, in our example, ends up only investigating transaction 6 as the result of their own model.

Figure 2 shows a potential issue with our pipeline: most banks in the IBM-AML dataset are small and process a small amount of transactions. It is not possible to train a machine learning model on such a tiny amount of transactions. Therefore, we need to create a new view of the IBM-AML dataset, which we call IBM-AML-LB (LB stands for ‘large banks’). IBM-AML-LB contains only transactions coming from the six banks which are involved in at least one million transactions in the large version of IBM-AML.

### 3.4. Methods

The method we apply to tackle the financial fraud detection problem is FraudGT [3]. We choose FraudGT because it is one of the top-performing models on the IBM-AML dataset.

FraudGT is one of the methods discussed in §2 that create a graph of financial transactions connecting accounts with the transactions flowing between them. It augments the graph with a number of techniques:

- *Reverse message passing* allows the convolution to go against the direction of the edge even in a directed network;
- *Port numbering* allows each node to track which one of its neighbours sent a specific message via the convolution layers;
- *Ego IDs* one-hot encode nodes so that the convolution is able to detect cycles.

As we show in the experiments, in the decentralized pipeline, a simpler approach can be preferred sometimes. We consider GatedGCN [38], an earlier iteration of FraudGT. GatedGCN shares many features and parameters with FraudGT but does not employ the three techniques we described here, making it a simpler baseline.

Both for FraudGT and for GatedGCN, we use the implementations provided by the authors of FraudGT.<sup>3</sup>

Since we have two steps (first banks, then authority) and two methods (FraudGT and GatedGCN), we have a total of four possible pipelines—as figure 4 shows. We can have the banks and the authority use the same method (either FraudGT or GatedGCN) or use different methods.

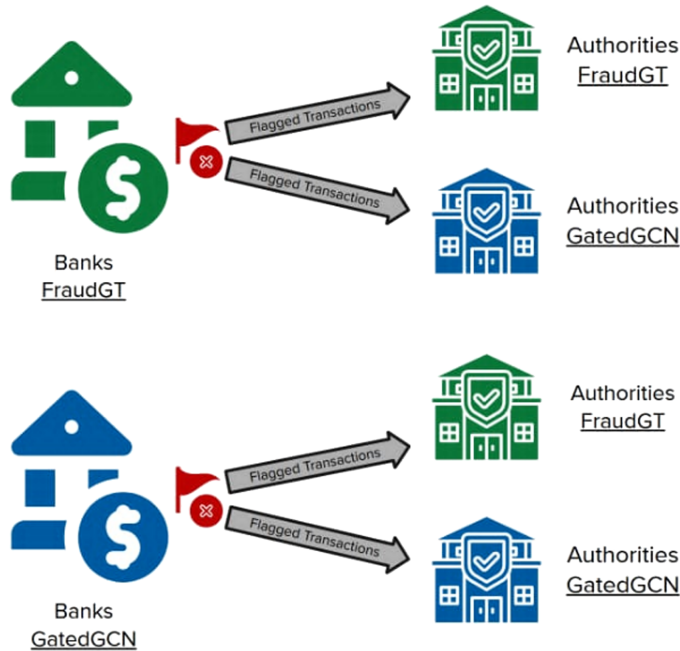
## 4. Experiments

### 4.1. Replication

#### 4.1.1. IBM-AML

We start evaluating our pipeline by replicating the results from the state-of-the-art. In this experiment, we run FraudGT on the full IBM-AML dataset with the same parameters from the original paper, using the original implementation. We perform this step to confirm that we are able to reproduce the original results and any performance difference we see hereafter is not due to an incorrect implementation.

<sup>3</sup><https://github.com/junhongmit/FraudGT>, date of access: January 30th, 2025.



**Figure 4.** The four possible combinations of methods in our two-step pipeline.

**Table 1.** The F1 scores for FraudGT on IBM-AML, as reported by the original paper and in our runs.

size	data	original F1	our F1
small	low	0.4701 $\pm$ 0.0222	0.4780
	high	0.7613 $\pm$ 0.0095	0.7629
medium	low	0.4406 $\pm$ 0.0527	0.4917
	high	0.7593 $\pm$ 0.0192	0.7711
large	low	0.3743 $\pm$ 0.0494	0.3746
	high	0.7334 $\pm$ 0.0164	0.7651

Table 1 lists the result. As we can see, we score higher than the reported values in the original paper in all cases, and in all cases, we are within the margin of error—except in large-high. We can conclude that we are able to properly reproduce the state-of-the-art.

Because we are specifically focusing on the  $s = 1$  class, we list its scores in table 2. Here we cannot compare with the original paper, since the original paper only reports the overall F1 score but, on account of table 1, we assume our evaluation is in line with the original performance. We omit reporting on the  $s = 0$  class, since the class imbalance and the sparsity of the output means that it will always be predicted with almost perfect accuracy (both precision and recall are 0.999). In other words, the vast majority of transactions are not fraudulent, and FraudGT only flags as fraudulent a vast minority of transactions.

When it comes to recall, we can see that the dataset versions with a higher prevalence of financial fraud are able to recover more than 70% of all fraudulent transactions. However, for the dataset with low amounts of fraud, we are never able to recover more than 40% of the fraudulent transactions. This is probably due to the fact that, in the low fraud prevalence data, there are not enough examples from the positive class to train properly.

## 4.1.2. IBM-AML-LB

To compare the traditional pipeline with a single central authority with our proposed two-step pipeline, we cannot use the state-of-the-art results with the IBM-AML dataset. This is because the two-step pipeline

**Table 2.** The precision, recall and F1 scores for FraudGT on IBM-AML for the  $s = 1$  fraud class.

size	data	precision	recall	F1
small	low	0.60	0.28	0.38
	high	0.85	0.74	0.79
medium	low	0.78	0.36	0.49
	high	0.79	0.72	0.75
large	low	0.48	0.23	0.31
	high	0.74	0.71	0.73

**Table 3.** The precision, recall and F1 scores for FraudGT and GatedGCN on IBM-AML-LB for the  $s = 1$  fraud class on the test set.

data	model	precision	recall	F1
high	FraudGT	0.63	0.83	0.71
	GatedGCN	0.37	0.55	0.44
low	FraudGT	0.51	0.23	0.32
	GatedGCN	0.55	0.16	0.24

needs to focus only on banks large enough to train the local bank model. As we explain in §3.3, we create the IBM-AML-LB dataset using the large IBM-AML data, selecting only the six largest banks processing more than one million transactions.

In this section, we implement the traditional pipeline by pooling together all six banks from IBM-AML-LB. The results of this section use exactly the same data as our two-step pipeline, and therefore the performances of the two pipelines are directly comparable.

Table 3 lists the results. We add the comparison with GatedGCN, as this method is important when considering the two-step pipeline. Here we can see two things. First, the performance of FraudGT does not change much when comparing the evaluation on the full IBM-AML large high/low—last two rows of table 2. This is expected as we are keeping a large fraction of the original dataset. Second, FraudGT is clearly superior to GatedGCN—confirming the results of the original paper.

## 4.2. Decentralized

### 4.2.1. Banks

In the two-step pipeline we propose, the first step is the bank step. Here, each bank trains, in isolation, their own model. Only the transactions that they flag as suspicious will be sent to the authority. Therefore, it is interesting to see how well in precision and recall these banks perform. If they achieve high precision, almost everything they send to the authorities is actual financial fraud. If they excel in recall, then few fraudulent transactions will be lost along the way, and most will reach the authority.

Table 4 lists the results of this step; three things stand out:

- The banks all more or less perform at the same level, with the notable exception of bank 70 (the largest bank) which does not flag any transaction as fraud;
- Just like in the previous pipeline, the low dataset is harder to handle as there are fewer transactions in the positive class to learn from;
- Between 20% and 30% of all fraudulent transactions are dropped in the high dataset.

The last point is particularly important because, by how recall works, the recall we achieve at the bank step effectively caps the recall of the whole pipeline. Even if the authority model could achieve perfect recall, it cannot retrieve fraudulent transactions that were not sent to them by the banks.

We can look at the performance of GatedGCN on the same task on the same data in table 4. Again, we confirm that GatedGCN does not perform as well as FraudGT in this task; however, it was able to achieve an excellent recall for bank 70 in the high version of the dataset. This high recall is helpful to give

**Table 4.** The precision, recall and F1 scores for FraudGT on IBM-AML-LB for the  $s = 1$  fraud class, per bank in the bank step.

data	model	bank	precision	recall	F1	
high	FraudGT	0	0.75	0.73	0.74	
		11	0.61	0.81	0.69	
		12	0.73	0.76	0.74	
		20	0.69	0.77	0.72	
		27	0.77	0.72	0.74	
		70	0.00	0.00	0.00	
		GatedGCN	0	0.56	0.77	0.65
	11	0.52	0.69	0.59		
	12	0.53	0.71	0.61		
	20	0.56	0.71	0.63		
	27	0.47	0.65	0.54		
	70	0.00	0.87	0.01		
	low	FraudGT	0	0.57	0.27	0.37
			11	0.34	0.38	0.36
12			0.42	0.37	0.39	
20			0.64	0.31	0.42	
27			0.15	0.18	0.17	
70			0.00	0.00	0.00	
GatedGCN			0	0.00	0.51	0.01
11		0.09	0.34	0.14		
12		0.34	0.24	0.28		
20		0.04	0.28	0.07		
27		0.16	0.12	0.14		
70		0.00	0.02	0.00		

a chance to the authority model to also achieve a high recall score. We now turn to the second step of the pipeline.

#### 4.2.2. Authorities

The authority collects all flagged transactions from all banks in the IBM-AML-LB dataset and runs its own model. As noted in §3.4, here we have four possible approaches, depending on whether the banks—and the authority—used different combinations of FraudGT or Gated GCN.

Table 5 lists the performances in all scenarios, for both the high and low variants of the dataset. When compared with table 3, we can see that applying the state-of-the-art FraudGT at both steps of the pipeline suffers a large penalty in recall—from 0.83 to 0.44 in the high data, 0.23–0.10 in the low data—which in turn penalizes F1 even with a higher precision. Remarkably, applying FraudGT at the bank level but GatedGCN at the authority level increases recall and F1 in both the high and low data. In general, using the most sophisticated and performant method in both steps in the pipeline leads to the *worst* performance, worse even than applying the simpler baseline in both steps. Applying GatedGCN in either step—preferably on the authority step—is always beneficial.

This is the key result of our paper, which is the need for banks and authorities to coordinate. The optimal scenario is to perform sophisticated analysis at the bank level and a simpler analysis at the authority level. However, if the authority thinks the banks run GatedGCN, table 5 shows that they should

**Table 5.** The precision, recall and F1 scores for the authority on IBM-AML-LB high for the  $s = 1$  fraud class on the test set.

data	bank	authority	precision	recall	F1
high	FraudGT	FraudGT	0.78	0.44	0.56
		GatedGCN	0.80	0.92	0.85
	GatedGCN	FraudGT	0.62	0.70	0.66
		GatedGCN	0.60	0.68	0.64
low	FraudGT	FraudGT	0.16	0.10	0.12
		GatedGCN	0.43	0.57	0.49
	GatedGCN	FraudGT	0.35	0.56	0.43
		GatedGCN	0.36	0.46	0.40

be running the sophisticated FraudGT instead, because it would perform better than running two GatedGCNs. However, some banks might be running FraudGT instead: if the authority does not know this and runs FraudGT, then they would have effectively implemented the worst possible pipeline, by acting rationally but with incomplete information.

#### 4.2.3. Real-world performance

While the performance on a framework should be judged on the test set as we have done so far, we should also be looking at the outcome of the pipeline on all transactions. This is in line with the message of this paper: we want to see a realistic outcome of what happens to all transactions, because the banks run their models on all transactions and send all suspicious transactions to the authorities, not just the test set. This is not done to evaluate the method, but to see how different pipelines might fail at different points for different reasons.

To do this study, it is convenient to start by looking at the confusion matrices, as they show what happens at each step of the pipeline. All sub-panels in [tables 6 and 7](#)—which refer to the high and low versions of the data—share the same temporal evolution from top to bottom: the first row contains the transaction the bank did not flag, the second the transactions the authority received and deemed not fraudulent, and the bottom row was the transactions flagged by the authority. The columns tell us the ground truth, whether the transaction was fraudulent (right) or not (left).

The general difference between FraudGT and GatedGCN is that the latter tends to favour precision over recall. FraudGT tends to be more restrictive and assigns more transactions to the  $s = 0$  class. This is evident in the high financial fraud scenario ([table 6](#)). Here, FraudGT stops at the first step 23 million transactions ([table 6a,b](#)), rather than the 6 million from GatedGCN ([table 6c,d](#)). This is an argument in favour of using FraudGT at the bank level: GatedGCN would send too many transactions to the authority, burdening it with a high computational load. This burden does not lead to an increase in recall, as the total number of correctly classified  $s = 1$  transactions is only 10 000 in [table 6c,d](#), while it is higher (12 000) in the FraudGT-GatedGCN pipeline ([table 6b](#)).

The situation is different in the low financial fraud scenario ([table 7](#)). Here, while still being more permissive at the bank step, GatedGCN does not swamp the authority with too many transactions: only 600 000 are sent to the second step ([table 7c,d](#))—which is still around 60 times more than what FraudGT flags at the bank step ([table 7a,b](#)). In this scenario, the best recall is achieved by applying FraudGT at the authority step after screening at the bank step using GatedGCN ([table 7c](#)).

In both scenarios, the two-step pipeline shows that a significant fraction of fraudulent transactions are lost at the bank step. This is confirmed by looking at the corresponding evaluation matrices. [Table 8](#) lists the evaluation measures when calculated on the full IBM-AML-LB dataset in the two-step pipeline, and not just on the test set as done in [table 5](#). Applying FraudGT twice leads to the lowest recall owing to its strictness.

[Table 8](#) should be compared with [table 9](#), which reports the evaluation for the one-step pipeline on all transactions (and not only on the test set as [table 3](#) does). We find results in line with the previous discussion: adding GatedGCN in some parts of the two-step pipeline is beneficial and leads to comparable performance with the one-step pipeline, even if in the one-step pipeline GatedGCN is clearly inferior to FraudGT in all scenarios.

**Table 6.** Confusion matrices for the pipelines in the high money laundering data. N/R stands for ‘not received’: the bank did not flag these transactions.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	23,113,344	38,438
	$s = 0$	4,035	8,683
	$s = 1$	1,250	3,772

(a) Bank model: FraudGT; Authority model: FraudGT.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	23,113,344	38,438
	$s = 0$	1,206	429
	$s = 1$	4,079	12,026

(b) Bank model: FraudGT; Authority model: GatedGCN.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	6,301,097	10,663
	$s = 0$	16,770,703	23,844
	$s = 1$	5,791	10,778

(c) Bank model: GatedGCN; Authority model: FraudGT.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	6,301,097	10,663
	$s = 0$	16,770,302	23,817
	$s = 1$	6,063	10,802

(d) Bank model: GatedGCN; Authority model: GatedGCN.

## 5. Conclusions

In this paper, we re-evaluate how performance in the financial fraud detection problem should be estimated. The classical approach in multi-bank data is to assume a single authority with perfect knowledge about all transactions in all banks, which is unrealistic. We propose a two-step pipeline, in which banks first flag their own transactions and send them to an authority, and then the authority pools all received transactions and trains its own model. We evaluate this specifically with the recall of the positive class, as we are interested in capturing as many fraudulent transactions as possible. Our results show that sophisticated methods which outperform simpler methods in the unrealistic one-step pipeline actually do not lead to the best performance when implementing this two-step pipeline. In our results, it always seems beneficial to inject in one step of the pipeline the simpler methods.

These results are important for the machine learning researchers, but also for the practitioners in the field. For the machine learning researchers, we provide evidence that a more realistic evaluation pipeline is necessary to properly evaluate new financial fraud detection methods. For the practitioners, we highlight the importance of bank-authority coordination. Naively applying the state-of-the-art methods at

**Table 7.** Confusion matrices for the pipelines in the low money laundering data.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	22,379,147	26,337
	$s = 0$	2,036	990
	$s = 1$	10,686	78

(a) Bank model: FraudGT; Authority model: FraudGT.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	22,379,147	26,337
	$s = 0$	11,985	269
	$s = 1$	737	799

(b) Bank model: FraudGT; Authority model: GatedGCN.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	21,754,034	25,606
	$s = 0$	638,180	535
	$s = 1$	997	946

(c) Bank model: GatedGCN; Authority model: FraudGT.

		True Label	
		$s = 0$	$s = 1$
Predicted	N/R	21,754,034	25,606
	$s = 0$	638,326	639
	$s = 1$	853	842

(d) Bank model: GatedGCN; Authority model: GatedGCN.

**Table 8.** The precision, recall and F1 scores for the authority on IBM-AML-LB high for the  $s = 1$  fraud class on all transactions in the dataset (training+test).

data	bank	authority	precision	recall	F1
high	FraudGT	FraudGT	0.75	0.07	0.13
		GatedGCN	0.75	0.24	0.36
	GatedGCN	FraudGT	0.65	0.24	0.35
		GatedGCN	0.65	0.24	0.35
low	FraudGT	FraudGT	0.01	0.00	0.00
		GatedGCN	0.52	0.03	0.06
	GatedGCN	FraudGT	0.49	0.03	0.07
		GatedGCN	0.50	0.03	0.06

**Table 9.** The precision, recall and F1 scores for FraudGT and GatedGCN on IBM-AML-LB for the  $s = 1$  fraud class on all transactions in the dataset (training+test).

data	model	precision	recall	F1
high	FraudGT	0.59	0.29	0.39
	GatedGCN	0.39	0.20	0.26
low	FraudGT	0.43	0.02	0.04
	GatedGCN	0.32	0.01	0.03

all steps is the natural rational choice when not knowing what is being applied in the other steps of the pipeline. However, our results show that this leads in the end to the *worst* possible performance.

Now that we have established the proof of concept for this pipeline, there is a need for further research. Here we focus on two methods, which cover the state-of-the-art for the approaches using GNNs. Comparing more methods—and their combinations—in this pipeline should be the logical next step. Eventually, this should lead to research that can optimize both steps of the pipeline, creating a new—*synergistic*—approach to tackling the financial fraud detection problem.

**Ethics.** This work did not require ethical approval from a human subject or animal welfare committee.

**Data accessibility.** The data source is an open dataset publicly available, created by IBM, and has not been preprocessed in any special way. Our code is also based on an already-shared open source GitHub repository: <https://github.com/junhongmit/FraudGT/>. The relevant code for this research work is stored on GitHub: [https://github.com/mikk-c/aml\\_pipeline](https://github.com/mikk-c/aml_pipeline) and has been archived within the Zenodo repository: [9].

**Declaration of AI use.** We have not used AI-assisted technologies in creating this article.

**Authors' contributions.** A.M.G.: conceptualization, data curation, investigation, methodology, software, validation, writing—original draft, writing—review and editing; L.B.A.: conceptualization, investigation, supervision, writing—original draft, writing—review and editing; M.C.: conceptualization, funding acquisition, project administration, supervision, visualization, writing—original draft, writing—review and editing.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

**Conflict of interest declaration.** We declare we have no competing interests.

**Funding.** This research was supported by the Pioneer Centre for AI, DNRF grant number P1.

## References

- Barone R, Masciandaro D. 2008 Worldwide anti-money laundering regulation: estimating the costs and benefits. *GBER* **10**, 243. (doi:10.1504/GBER.2008.019983)
- Hilal W, Gadsden SA, Yawney J. 2022 Financial fraud: a review of anomaly detection techniques and recent advances. *Expert Syst. Appl.* **193**, 116429. (doi:10.1016/j.eswa.2021.116429)
- Lin J, Guo X, Zhu Y, Mitchell S, Altman E, Shun J. 2024 FraudGT: a simple, effective, and efficient graph transformer for financial fraud detection. In *ICAIIF '24*, Brooklyn NY USA, pp. 292–300. New York, NY, USA. (doi:10.1145/3677052.3698648)
- Fan J. 2025 Deep learning approaches for anti-money laundering on mobile transactions: review, framework, and directions. *arXiv Preprint arXiv arXiv:2503.10058*.
- Arjun S, Devanand K, Mahesh A. 2025 A multi-expert transformer model for click fraud detection. In *2025 5th international conference on pervasive computing and social networking (ICPCSN)*, pp. 1474–1480. New York, NY: IEEE.
- Altman E, Anghel A, Atasu K, Blanuša J, Egressy B, Von Niederhäusern L. 2023 Realistic synthetic financial transactions for anti-money laundering models. *Adv. Neural Inf. Process. Syst.* **36**, 29851–29874. (doi:10.52202/075280-1300)
- Rosen LW, Miller RS. 2022 *The financial crimes enforcement network (fincen): anti-money laundering act of 2020 implementation and beyond*. Congressional Research Service.
- Jensen TL, Martinello A, Mønsted BM. 2021 *Databaseret indsats styrker kampen mod hvidvask*. Danmarks Nationalbank.
- adagige. 2026 mikk-c/aml\_pipeline: Code Deposition for Reproducibility (reproducibility). Zenodo (doi:10.5281/zenodo.18934619)
- Weber M. 2018 Scalable graph learning for anti-money laundering: a first look. *arXiv:1812.00076*
- Jensen RIT, Iosifidis A. 2023 Fighting money laundering with statistics and machine learning. *IEEE Access* **11**, 8889–8903. (doi:10.1109/ACCESS.2023.3239549)
- Weber M. 2019 Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY: ACM.
- Alotibi J, Almutanni B, Alsubait T, Alhakami H, Baz A. 2022 Money laundering detection using machine learning and deep learning. *IJACSA* **13**, 732–738. (doi:10.14569/IJACSA.2022.0131087)

14. Alarab I, Prakoonwit S. 2023 Graph-based LSTM for anti-money laundering: experimenting temporal graph convolutional network with bitcoin data. *Neural Process. Lett.* **55**, 689–707. (doi:10.1007/s11063-022-10904-8)
15. Oliveira C. 2021 GuiltyWalker: distance to illicit nodes in the bitcoin network. *arXiv Preprint arXiv:2102.05373*.
16. Tatulli MP. 2023 HAMLET: a transformer based approach for money laundering detection. In *International symposium on cyber security, cryptology, and machine learning*, pp. 234–250. London, UK: Springer. (doi:10.1007/978-3-031-34671-2\_17)
17. Kute DV, Pradhan B, Shukla N, Alamri A. 2024 Explainable deep learning model for predicting money laundering transactions. *International Journal on Smart Sensing and Intelligent Systems* **17**. (doi:10.2478/ijssis-2024-0027)
18. Chen Z, Soliman WM, Nazir A, Shorfuzzaman M. 2021 Variational autoencoders and wasserstein generative adversarial networks for improving the anti-money laundering process. *IEEE Access* **9**, 83762–83785. (doi:10.1109/ACCESS.2021.3086359)
19. Ebiaredoh-Mienye SA, Esenogho E, Swart TG. 2021 Artificial neural network technique for improving prediction of credit card default: a stacked sparse autoencoder approach. *IJECE* **11**, 4392. (doi:10.11591/ijece.v11i5.pp4392-4402)
20. Mienye ID, Esenogho E, Modisane C. 2025 Detecting imbalanced credit card fraud via hybrid graph attention and variational autoencoder ensembles. *AppliedMath* **5**, 131. (doi:10.3390/appliedmath5040131)
21. Esenogho E, Mienye ID, Swart TG, Aruleba K, Obaido G. 2022 A neural network ensemble with feature engineering for improved credit card fraud detection. *IEEE Access* **10**, 16400–16407. (doi:10.1109/ACCESS.2022.3148298)
22. Liu Y, Ao X, Qin Z, Chi J, Feng J, Yang H, He Q. 2021 Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *WWW '21, Ljubljana Slovenia*, pp. 3168–3177. New York, NY, USA. (doi:10.1145/3442381.3449989)
23. Starnini M *et al.* 2021 Smurf-based anti-money laundering in time-evolving transaction networks. In *ECML pkdd*, pp. 171–186. London, UK: Springer. (doi:10.1007/978-3-030-86514-6\_11)
24. Eddin AN. 2021 Anti-money laundering alert optimization using machine learning with graphs. *arXiv Preprint arXiv:2112.07508*.
25. Cardoso M, Saleiro P, Bizarro P. 2022 Laundrograph: self-supervised graph representation learning for anti-money laundering. In *ICAIF '22*, pp. 130–138. New York, NY, USA. (doi:10.1145/3533271.3561727)
26. Cheng D, Ye Y, Xiang S, Ma Z, Zhang Y, Jiang C. 2023 Anti-money laundering by group-aware deep graph learning. *IEEE Trans. Knowl. Data Eng.* **35**, 12444–12457. (doi:10.1109/TKDE.2023.3272396)
27. Barone M, Coscia M. 2018 Birds of a feather scam together: trustworthiness homophily in a business network. *Soc. Networks* **54**, 228–237. (doi:10.1016/j.socnet.2018.01.009)
28. Oztas B, Cetinkaya D, Adedoyin F, Budka M, Dogan H, Aksu G. 2023 Enhancing anti-money laundering: development of a synthetic transaction monitoring dataset. In *2023 IEEE International Conference on e-Business Engineering (ICEBE)*, Sydney, Australia, pp. 47–54. (doi:10.1109/ICEBE59045.2023.00028)
29. Jensen RIT, Ferwerda J, Jørgensen KS, Jensen ER, Borg M, Krogh MP, Jensen JB, Iosifidis A. 2023 A synthetic data set to benchmark anti-money laundering methods. *Sci. Data* **10**, 661. (doi:10.1038/s41597-023-02569-2)
30. Girish KK, Bhowmik B. 2024 Money laundering detection in banking transactions using RNNs and hybrid ensemble. In *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kamand, India, pp. 1–7. (doi:10.1109/ICCCNT61001.2024.10724384)
31. Luo X, Han X, Zuo W, Wu X, Liu W. 2024 A semi-supervised money laundering detection framework based on decoupling training. *IEEE Trans. Inform. Forensic Secur.* **19**, 4518–4533. (doi:10.1109/TIFS.2024.3380262)
32. Egressy B, Von Niederhäusern L, Blanuša J, Altman E, Wattenhofer R, Atasu K. 2024 Provably powerful graph neural networks for directed multigraphs. *AAAI* **38**, 11838–11846. (doi:10.1609/aaai.v38i10.29069)
33. Xu K. 2018 How powerful are graph neural networks? *arXiv Preprint arXiv:1810.00826*.
34. Zhang C, Xie Y, Bai H, Yu B, Li W, Gao Y. 2021 A survey on federated learning. *Knowl. Based Syst.* **216**, 106775. (doi:10.1016/j.knosys.2021.106775)
35. Wen J, Zhang Z, Lan Y, Cui Z, Cai J, Zhang W. 2023 A survey on federated learning: challenges and applications. *Int. J. Mach. Learn. Cybern.* **14**, 513–535. (doi:10.1007/s13042-022-01647-y)
36. Suzumura T. 2022 *Federated Learning for collaborative financial crimes detection*, pp. 455–466. Cham: Springer International Publishing. (doi:10.1007/978-3-030-96896-0\_20)
37. Guembe B, Azeta A, Osamor V, Ekpo R. 2023 A federated machine learning approaches for anti-money laundering detection. *SSRN Journal* (doi:10.2139/ssrn.4669561)
38. Cao S. 2019 Titant: online real-time transaction fraud detection in ant financial. *arXiv Preprint arXiv:1906.07407*. (doi:10.14778/3352063.3352126)