THE GANFATHER: CONTROLLABLE GENERATION OF MALICIOUS ACTIVITY TO EXPOSE DETECTION WEAKNESSES AND IMPROVE DEFENCE SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose *The GANfather*, an adversarial method without label requirements, to enhance the detection of illicit activity in various domains. It comprises a generator that produces meaningful attacks and a discriminator to detect them. *The GANfather* differs from anomaly detection as it generates attacks mimicking legitimate activity. It also differs from GANs as it focuses on generating out-of-sample data, namely examples of a novel class, such as illicit activity. We employ hyperparameter tuning to balance both goals. Optionally, we may encourage the generator to bypass pre-existing detection systems. This setup reveals defensive weaknesses for the discriminator to correct.

We evaluate our method in two real-world use cases, money laundering, and recommendation systems. In the first, an estimated 0.7–3 trillion euros are laundered annually undetected. Our method moves cumulative amounts close to 250 thousand dollars through a network of accounts without being detected by existing systems. In the latter, we recommend the target item to a broad user base with as few as 30 synthetic attackers, corresponding to 0.5% of total users. In both cases, we train a new defence system to capture the synthetic attacks, potentially saving millions of dollars in detected criminal activity. Finally, our method is general and applies to other use cases targeted by adversarial attacks.

1 INTRODUCTION

Illicit activities frequently target digital systems and services. Importantly, these illicit activities are adversarial: an attacker and a defence system constantly adapt to each other's behaviour.

For instance, the rise of digital banking enables clients to open bank accounts and perform financial transactions efficiently worldwide but enables complex money laundering schemes.Recent estimates indicate undetected money laundering activities of €0.7–3 trillion annually (Lannoo & Parlour, 2021) and operational costs of $37.1 billion incurred by financial institutions (Ray, 2021).

Recommender systems provide another example. They are often embedded in digital services to deliver personalisation, matching users with content, goods, services, or people based on their interests. However, recommenders may suffer injection attacks whenever malicious actors fabricate signals (e.g., clicks, ratings, or reviews) to influence recommendations. The consequences of these distortions are socially and economically meaningful. For example, a one-star decrease in restaurant ratings can lead to a 5 to 9 percent decrease in revenue (Luca, 2016)[1]. Detecting malicious agents is far from trivial. A critical challenge relates to class imbalance, as illicit activity is rare. Additionally, labelled datasets are often unavailable or incomplete due to the absence of natural labels and the cost of feedback, primarily generated through manual labelling. Even when labels are available, the feedback is not immediate, as investigations are far from trivial. Furthermore, illicit agents in-

[1]Many other examples exist where malicious agents continuously adapt to exploit defences. Email providers constantly battle spammers. Social networks and messaging apps must deal with abusive content and the spread of misinformation, for example, through *bot accounts*. Search engines face scammers attempting to use them to target potential victims. These illicit activities have a meaningful impact and force digital services to incur severe operational costs to mitigate them.

tentionally mimic legitimate activity to avoid detection. Finally, due to the adversarial environment, malicious agents continuously adapt existing strategies, resulting in distribution shift.

We propose *The GANfather*, a method to generate examples of illicit activity and train effective detection systems without any labelled examples leveraging Generative Adversarial Networks (GANs). We train a model to discriminate between real and synthetic data while training a generator to fool it by learning to generate realistic data (Goodfellow et al., 2014). Unlike typical GANs, however, our goal is not to learn the real data distribution but out-of-distribution examples representing meaningful attacks.

Our method includes an additional optimisation objective in the training loss of the generator to generate meaningful attacks. This objective is a use-case-specific, user-defined differentiable formulation of the goal of the malicious agents, i.e., what they are optimising. For example, the objective of an injection attack in a recommender system may be to increase the frequency of recommendation of a given item. Alternatively, the optimisation objective in money laundering may be how much money flows through certain accounts, a practice commonly known as layering, to conceal the criminal origin of the funds.

Furthermore, our method may consider an existing defence system as long as a differentiable formulation is possible. In that case, we reward the generator for not triggering existing detection mechanisms. Thus, our method can actively find liabilities in the current system while simultaneously training a complementary detection system to correct them.

We summarise our main contributions as follows:

- **Automated generation of attacks:** We propose a framework to generate synthetic data, simulating adversarial attacks by malicious actors, where the distance between real and synthetic distributions is controllable by hyperparameters.
- **Detection of attacks:** We simultaneously train a detection system, distinguishing the synthetic attacks from the real data.
- **No labels:** We do not require any labelled examples of malicious activity; the method can devise previously unseen, unlabelled attacks. Instead, we generate attacks using a differentiable formulation of the intended behaviour. Our method is suitable for creating examples of a class for which we do not have data.
- **Expose and augment current defence system:** Optionally, if a detection system is in place and it is possible to formalise it in a differentiable manner, we may train the generator to bypass it. Additionally, we may use the resulting discriminator as a complementary model, specialised in detecting attacks that evade the existing system.
- **Our method is general:** We validate and provide an experimental study of our method in two domains: money laundering and recommendation systems.

The remainder of the paper is structured as follows. We first describe our proposed framework in Section 2. We detail our experiments and discuss results in Section 3. We provide an overview of related work in Section 4. Finally, we present our conclusions in Section 5.

2 METHODS

This section provides a general description of our proposed framework in Section 2.1. We proceed to describe two use-cases: AML (Section 2.2) and injection attacks in recommendation systems (Section 2.3).

2.1 GENERAL DESCRIPTION

Figure 1 depicts the general structure of our framework. It includes a generator (Section 2.1.1), a discriminator (Section 2.1.2, an optimisation objective (Section 2.1.3), and, optionally, an existing alert system (Section 2.1.4).

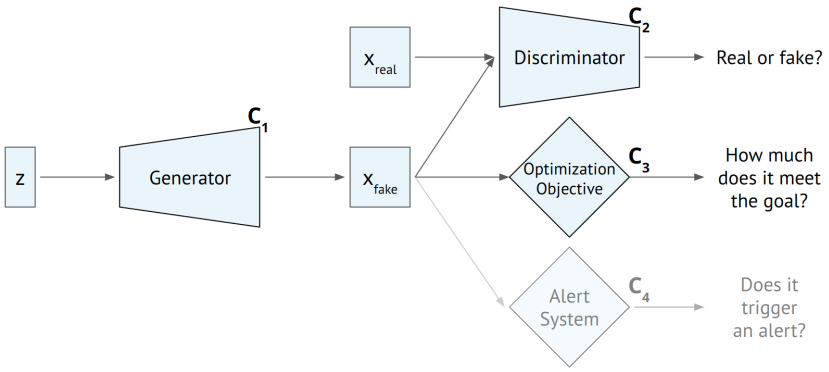


Figure 1: *The GANfather* framework. Its main components comprise a generator, C_1, which generates realistic attacks, and a discriminator, C_2, which detects these attacks. Our method includes an optimization objective, C_3, to incentivise realistic instances. Finally, our method supports the inclusion of an existing alert system, C_4.

2.1.1 GENERATOR

As in the classical GAN architecture, the generator receives a random noise input vector and outputs an instance of data. However, unlike classical GANs, its output is passed to multiple components, which will return feedback as a gradient during training. The loss of the generator is a linear combination of the losses of the components, where the weights are hyperparameters controlling the relative strength of the various losses. It contains the following terms:

$$\mathcal{L}(G) = \alpha\mathcal{L}_{Obj}(G, O) + \beta\mathcal{L}_{GAN}(G, D) + \gamma\mathcal{L}_{Alert}(G, A) \tag{1}$$

where $\mathcal{L}_{Obj}(G, O)$ denotes the optimisation objective; $\mathcal{L}_{GAN}(G, D)$ denotes the GAN loss; $\mathcal{L}_{Alert}(G, A)$ is an optional term representing an existing detection system; and α, β, and γ are hyperparameters to tune the strength of each component.

2.1.2 DISCRIMINATOR

The discriminator receives an example and produces a score indicating the likelihood that the example is real or synthetic. It serves two primary purposes. We assume malicious agents mimic legitimate users; therefore, the generator's synthetic examples should be similar to real data. The discriminator's feedback will nudge the generator towards realistic examples. Moreover, the discriminator eventually learns to distinguish synthetic attacks from real data, which may function as an automatic detection system for illicit activity. Notably, the discriminator has no label requirements since it learns to detect illicit activity based on the generated synthetic attacks. We use the Wasserstein loss (Arjovsky et al., 2017) as our GAN loss.

2.1.3 OBJECTIVE

The optimisation objective quantifies how well the synthetic example is fulfilling the goal of a malicious agent. It can be a mathematical formulation or a differentiable model of the goal. Suppose we train the generator to optimise the output of the objective function. In that case, the synthetic data approximates the illicit behaviour, as it will actively learn strategies aligned with the optimisation objective. This allows the generator to find previously unseen strategies to meet malicious goals.

2.1.4 ALERT SYSTEM

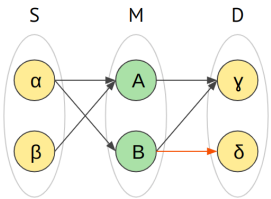
If an existing, differentiable alert system is present, we can add it to our framework to teach the generator to create examples that do not trigger detection. Naturally, this is the most realistic scenario in adversarial environments, as malicious agents actively try to bypass detection in real life. Furthermore, it is beneficial for the discriminator to focus on undetected illicit activity, so we include

the alert system in the generator's loss such that it can learn to avoid triggering it. Whenever the existing system is not differentiable, a differentiable proxy may be possible.

2.2 Anti-Money Laundering

We tackle the layering stage of money laundering, in which criminals attempt to conceal the origin of the money by moving large amounts across financial institutions through what is known as "mule accounts", creating a flow of money from the perspective of each institution. To represent these transactions, we can use a 3D tensor as depicted in Figure 2. The first two dimensions correspond to the weighted adjacency matrix of the accounts and the third dimension is time. We discretise the events into time windows of fixed length and group events that belong to the same entry in the tensor by summing their amounts. Our representation covers any dynamic network with a 3D tensor whose size is fixed and pre-specified, which allows us to avoid recurrent models. While this approach may limit the size of generated data, domain experts reported that up to 95% of the money-laundering investigations involve cases containing up to 5 accounts.

Source	Target	Amount	Day
α	A	3.14	1
α	B	15.92	2
β	A	65.35	2
A	γ	89.79	1
B	γ	32.38	1
B	δ	46.26	2
B	δ	43.38	2



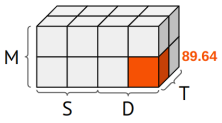


Figure 2: Data representation of transactional data. From the raw tabular data, we build the tripartite graph of the transactions, which is in turn represented as a 3D tensor.

2.2.1 Generator

We implement the generator (Section 2.1.1) using a set of dense layers, followed by a set of transposed convolutions. Then, we create two branches: one to generate transaction amounts and the other to generate transaction probabilities. We use the probabilities to perform categorical sampling and generate sparse representations, similar to real transaction data. After the sampling step, the two branches are combined by element-wise multiplication, resulting in a final output tensor with the dimensions described above. More details of the generator's architecture are found in Table 1.

2.2.2 Discriminator

The discriminator (Section 2.1.2) receives two tensors with the same shape as inputs: one containing the total amount of money transferred per entry, and the other with the count information (mapping positive amounts to 1 and empty entries to 0). Each tensor passes through convolutional layers, followed by permutation-invariant operations over the internal and external accounts. Then, we concatenate both tensors. We reduce the dimensionality of the final vector to a classification outcome using dense layers. More details of the discriminator architecture are found in Table 2.

2.2.3 Objective

To characterise the money flow behaviour of layering, we define the objective function (Section 2.1.3) as the geometric mean of the total amount of incoming and outgoing money per internal account (Equation 2). This objective function incentivizes the generator to increase the amount of money sent and received per account and keep these two quantities similar.

$$R_{aml}(x) = \sum_{i=0}^{M} \sqrt{(\mathcal{S}_{in}(x[i]) \times \mathcal{S}_{out}(x[i])} \tag{2}$$

2.2.4 ALERT SYSTEM

In AML, it is common to have rule-based detection systems. However, rules are usually not differentiable, and our generator requires feedback in the form of a gradient. Hence, we construct a deep learning model as a proxy for the rules system. We train this model before the other framework components, using augmented real data and the rule's predictions for each example as labels. This proxy network will then give the same feedback as the rules system, but in a differentiable way, so we can backpropagate its feedback to the generator.

2.3 RECOMMENDATION SYSTEM

In this work, we consider collaborative filtering recommender systems. However, our method is compatible with any other differentiable recommender. The system receives a matrix of ratings R with shape (Nu, Ni), where Nu is the number of users and Ni is the number of items. First, we compute the matrix D of shape (Nu, Nu) with the cosine distance between the users. Then, we compute the predicted ratings P as a dot product between D and R [2].

2.3.1 GENERATOR

The generator consists of multi-layer perceptrons that gradually increase the size of the random noise input vector to the output vector of ratings. For gradients to flow better, we implement the network with residual blocks as in ResNet (He et al., 2016), but using two dense layers instead of the convolutional layers. We use residual blocks to process the input noise vector. Then, similarly to the AML implementation, we create two branches: one for ratings and the other for probabilities. Each of these passes through additional residual blocks until the last dense layer, where we scale up the vector size to the number of items Ni, before doing the categorical sampling step. Importantly, each synthetic user is independent, but the architecture could easily be adapted to generate them together. More details of the generator architecture available in Table 4.

2.3.2 DISCRIMINATOR

The architecture of the discriminator mirrors that of the generator since each user is also processed independently. It receives two tensors with the same shape as inputs: one containing the ratings and the other with the count information. We scale down each tensor with a dense layer before passing through the residual blocks. Finally, we concatenate the two vectors into a single vector. After passing it through additional residual blocks, we scale down the final vector to single value output with a dense layer. Invariant permutation is not required because we generate and process each user separately. More details of the discriminator architecture available in Table 5.

2.3.3 OBJECTIVE

We define the goal of malicious agents as increasing the frequency of recommendation of a specific item. The objective function in Equation 3 incentivizes the generator to increase the rating of the target item for every user.

$$R_{rs}(P) = -\sum_{i=0}^{Nu}\sum_{j=0}^{Nm} ReLU(P[i,j] - P[i,t]) \tag{3}$$

3 RESULTS

We evaluate the efficiency of *TheGANfather* to generate attacks and also to detect them in two use-cases: money laundering (Section 3.1) and recommendation system (Section 3.2).

[2] We decide not to represent time since most classical recommender systems do not account for it. However, it is possible to temporal information using a similar setup to what we described in the AML use case.

3.1 MONEY LAUNDERING

3.1.1 OVERVIEW

We use a real-world dataset of financial transactions, containing approximately 200000 transactions, between 100000 unique accounts, over 10 months. We implement *The GANfather*'s generator and discriminator following the architectures presented in Table 1 and Table 2, respectively. The rules detection system contains five scenarios, capturing known suspicious patterns such as a sudden change in behaviour or rapid movements of funds[3]. Instead of pre-training the rules detection system, we hard-code the rules and implement the logical gates using arithmetic operators. We conduct a hyperparameter search over α, β and γ (discussed in Section 2.1.1) to study the impact of different the relative weights for each component, as shown in Table 3.

3.1.2 RESULTS

We perform a random search over the hyperparameters in Table 3. In Figure 3, we show a projection of the various generators trained with different hyperparameters. The horizontal and vertical axes represent multi-dimensional scaling of the distance of amount distribution and probability distribution, respectively. We depict the amount of money flowing using colour. The size indicates the diversity score (Equation 4). Figure 3 shows that the generators trained to avoid both the existing detection system and the discriminator can move up to 250k dollars through just five accounts. We also observe that generators with higher diversity scores typically are the ones that circulate more money while avoiding detection.

Next, we build a dataset combining real and synthetic data. We ensure that the synthetic data contains a variety of behaviours by sampling from various generators at various epochs during training and with different random noise seeds. We use this dataset of synthetic examples to evaluate snapshots of the trained discriminators and quantify how well they detect the synthetic data (Figure 4). We observe that the discriminator typically achieves a perfect classification performance, especially for higher values of the β parameter.

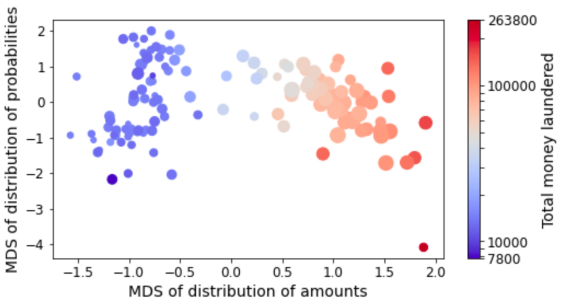


Figure 3: Hyperparameter search results. Each point corresponds to a different generator.

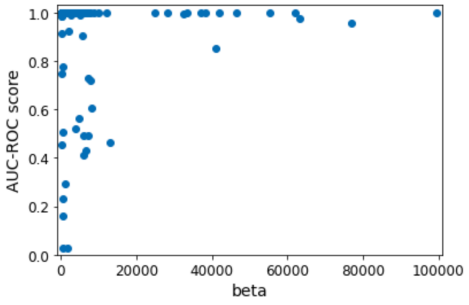


Figure 4: Discriminator AUC. Each point corresponds to a different discriminator.

These experiments show that *The GANfather*'s generator moves high volumes of money through a small set of accounts without triggering the alert system. Furthermore, *The GANfather*'s discriminator can complement the rules system and allows us to detect illicit activity undetected by the rules system consistently.

3.2 RECOMMENDER SYSTEM

3.2.1 OVERVIEW

We use the MovieLens 1M dataset, comprised of a matrix of $6,040$ users and $3,706$ movies, with ratings ranging from 1 to 5 (Harper & Konstan, 2015). We implement the generator and discriminator according to Table 4 and Table 5, respectively. We use the collaborative filtering recommender

[3]The AML rules consist of logical conditions comprised of a linear combination of profiles and thresholds.

system described in Section 2.3. During training, we take a weighted average of ratings considering all users in the dataset. In contrast, we only consider the top-400 closest neighbours for inference since we observed empirically that this value produces the best recommendation loss. We consider all users during training because the initially generated ratings are random, and only providing feedback from the top-400 closest users limits the strategies that the generator can learn. We assume that there is no previous detection system in place. We train our networks with 300 synthetic attackers but evaluate the generator's ability to influence the recommender system with various injection attacks. We define four baseline attacks: (1) a rating of 5 for the target movie and 0 otherwise, (2) a rating of 5 for the target movie and $\sim$90 random ratings for randomly chosen movies, (3) a rating of 5 for the target movie and $\sim$90 random ratings for the top 10% highest rated movies, (4) a rating of 5 for the target movie and $\sim$90 random ratings for the top 10% most rated movies.

3.2.2 Results

We perform an extensive hyperparameter search for α. In Figure 5, we observe that the generated attacks consistently outperform the baselines in recommending the target movie, with as few as 30 synthetic attackers, corresponding to 0.5% of the number of real users. Furthermore, increasing α leads to generators whose attacks make the recommender system recommend the target movie to an increasingly large portion of the real users, at the cost of moving further away from the real data distribution (measured through the KL divergence). Increasing the number of generated users also increases the target movie's recommendation frequency to real users.

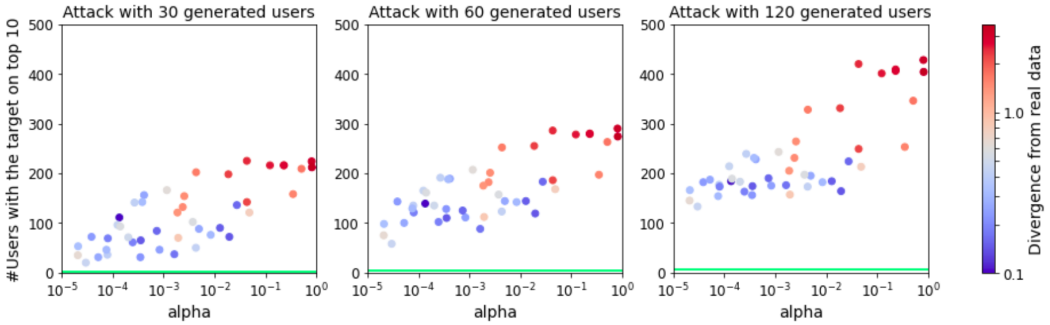


Figure 5: Hyperparameter search results. Each point corresponds to a different generator. The x-axis corresponds to the alpha value, the y-axis to the number of users with the target item on their top-10, and the colour to the KL divergence between real and generated data. We include the number of generated users above each panel. The green line represents the best baseline in each panel.

We subsequently analyse the generated attacks. In Figure 6, we see that the generated attacks overlap with real users more than the naive baselines, as expected in collaborative filtering. Furthermore, we see in Figure 7 that several generators are closer to real data than the baselines.

Finally, we analyse the detection of synthetic attacks. We build a dataset containing real and synthetic data. We then quantify the AUC of the pre-trained discriminators. In Figure 8, we observe that most discriminators achieve around 0.75 AUC. Then, we test training a discriminator directly on this synthetic dataset, obtaining near-perfect classification (Figure 9).

4 Related Work

4.1 Controllable data generation

Wang et al. (2022) review controllable data generation with deep learning. Among the presented works, we highlight De Cao & Kipf (2018). It leverages a GAN trained with reinforcement learning to generate small molecular graphs that verify desired properties. Their work is similar to ours in that we both (1) train a generator with a discriminator and a reward function to generate data and (2) use similar data representations, namely sparse matrices. However, we differ in motivation and goal. Whereas De Cao & Kipf (2018) concerns generating realistic data that verifies some conditions (e.g.,

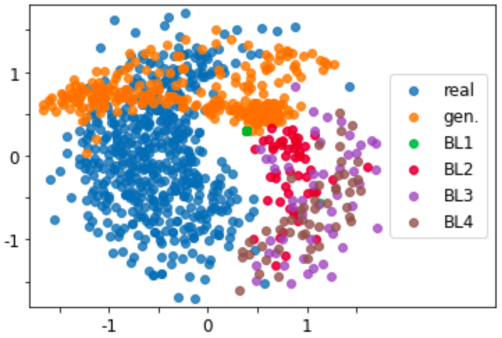


Figure 6: Projection of multi-dimensional scaling of user embeddings.

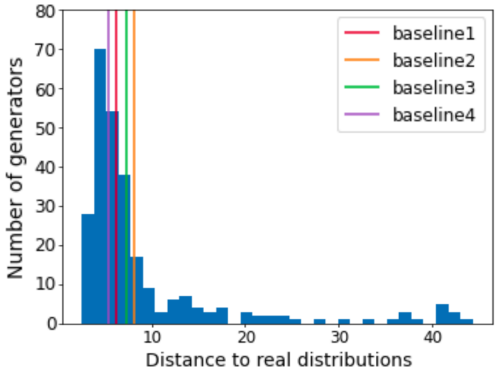


Figure 7: Histogram of distances between the distributions of generated attacks and real data.

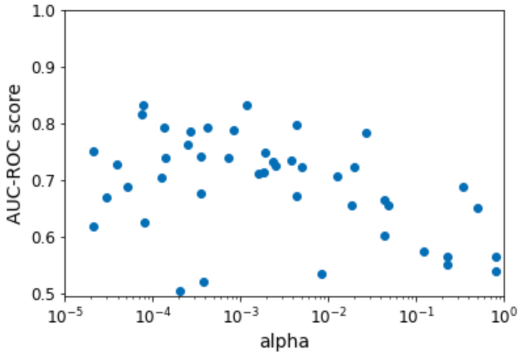


Figure 8: Discriminator AUC. Each point corresponds to a different discriminator.

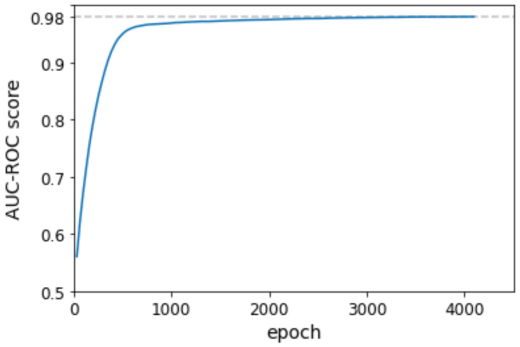


Figure 9: We can train a discriminator with almost perfect performance.

as our method could achieve leveraging the optional alert system), *The GANfather* generates out-of-distribution data to tackle adversarial domains. We address this issue by adding a novel component: the optimisation objective.

4.2 Anti-Money Laundering

Typical anti-money laundering solutions are rule-based (Watkins et al., 2003; Savage et al., 2016; Weber et al., 2018). They encode domain knowledge, ensure compliance, and generate interpretable alerts. However, rules suffer from high false-positive rates, may fail to detect complex schemes, and are costly to maintain. Machine learning-based solutions tackle these problems (Chen et al., 2018). Given the lack of labelled data, most solutions employ unsupervised methods like clustering (Wang & Dong, 2009; Soltani et al., 2016), and anomaly detection (Gao, 2009; Camino et al., 2017). These assume that illicit behaviours are rare and distinguishable, which may not hold whenever money launderers mimic legitimate behaviour.

Supervised methods have also been explored, including Random Forests (Jullum et al., 2020), Bayesian Networks (Raza & Haider, 2011), Neural Networks (Lv et al., 2008), and SVMs (Tang & Yin, 2005). In practice, anti-money laundering labels are typically incomplete, delayed, and expensive. Thus, most of these works assume that real data correspond to legitimate activity and use synthetic positive examples for training. Alternatively, Deng et al. (2009) and Lorenz et al. (2020) propose efficient label collection with active learning (Charitou et al., 2021) explores data augmentation using conditional GANs. We propose learning a generator through adversarial training, combined with an optimisation objective and, optionally, an existing detection system, to produce the synthetic positive instances.

There is a growing body of work on leveraging network properties and graph information to improve detection. Proposed techniques include feature engineering (Oliveira et al., 2021), recursive neural networks (RNNs) (Branco et al., 2020), and graph convolutional networks (GCNs) (Weber et al., 2019). Our method, alternatively, relies on a fixed, pre-specified structure to represent relational information while avoiding the need for sequential or graph-based methods.

Lastly, Li et al. (2020) and Sun et al. (2021) identify dense flows in large transaction graphs. Although we use similar representations, in our work, the representation underlies the generation of illicit money flows while training a discriminator for detection.

4.3 Recommender systems injection attacks

Most injection attacks on recommender systems are hand-crafted according to simple heuristics. Examples include random and average attacks (Lam & Riedl, 2004), bandwagon attacks (Burke et al., 2005a) and segmented attacks (Burke et al., 2005b). However, these strategies are less effective and easily detectable as most generated profiles differ significantly from real data and correlate with each other. Tang et al. (2020) address the optimisation problem of finding the generated profiles that maximise their goals directly through gradient descent and a surrogate recommender system. Alternatively, certain attacks target specific types of recommender systems, including association rules (Yang et al., 2017), matrix factorisation (Fang et al., 2020), and graph-based recommenders (Fang et al., 2018).

The detection of injection attacks may consist of statistical monitoring Chirita et al. (2005). More advanced detection methods include outlier detection (Chakraborty & Karforma, 2013; Davoudi & Chatterjee, 2017), SVMs (Zhang & Zhou, 2014), neural networks (Ebrahimian & Kashef, 2020; Meleshko et al., 2020), and among others (He et al., 2010; Xia et al., 2015). In recent years, some studies apply GANs to recommender systems to generate attacks and defend the system. Wu et al. (2021) combines a graph neural network (GNN) with a GAN to generate their attack. The former select which items to rate, and the latter decides the ratings. Zhang et al. (2021) and Lin et al. (2022) propose a similar setup to ours in which they train a GAN to generate realistic data and add a loss function to guide the generation of profiles to perform malicious recommendations as expected. Our work differs from theirs in two key aspects. Firstly, we use a different objective function. Secondly, we generate profiles directly from the noise vector. We then use categorical sampling to make our output sparse, while they base their profile generation on real data to obtain sparse ratings, as proposed in Chae et al. (2018)).

5 Conclusion

In this work, we propose *The GANfather* to generate data of a novel class without labelled examples, while simultaneously training a detection network to classify the novel class correctly.

Our strategy differs significantly from the existing anomaly detection literature as our generator produces data close to real data. Additionally, it differs from traditional GANs as we focus on generating out-of-sample data. The objective function nudges the generator from merely replicating the distribution of real data, instead balancing the objective with the similarity to real data through hyperparameter tuning. Thus, we can generate data ranging from likely legitimate to likely malicious activity.

We successfully apply our method to two use-cases: money laundering and recommendations. In both scenarios, a defence system has to adapt to the evolving behaviour of attackers. We show that our method generates attacks that bypass in-place systems but the proposed discriminator.

Our method fits the adversarial game between criminals and security systems by simulating various meaningful attacks. If existing defences are in place, our method may learn to avoid them and, eventually, train a complementary model. We hope our work contributes to increase robustness against illicit actors.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana SC Almeida, João Tiago Ascensão, and Pedro Bizarro. Interleaved sequence rnns for fraud detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3101–3109, 2020.

Robin Burke, Bamshad Mobasher, and Runa Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of 3rd international workshop on intelligent techniques for web personalization (ITWP 2005), 19th international joint conference on artificial intelligence (IJCAI 2005)*, pp. 17–24, 2005a.

Robin Burke, Bamshad Mobasher, Runa Bhaumik, and Chad Williams. Segment-based injection attacks against collaborative filtering recommender systems. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 4–pp. IEEE, 2005b.

Ramiro Daniel Camino, Radu State, Leandro Montero, and Petko Valtchev. Finding suspicious activities in financial transactions and distributed ledgers. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 787–796. IEEE, 2017.

Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 137–146, 2018.

Parthasarathi Chakraborty and Sunil Karforma. Detection of profile-injection attacks in recommender systems using outlier analysis. *Procedia Technology*, 10:963–969, 2013.

Charitos Charitou, Simo Dragicevic, and Artur d'Avila Garcez. Synthetic data generation for fraud detection using gans. *arXiv preprint arXiv:2109.12546*, 2021.

Zhiyuan Chen, Ee Na Teoh, Amril Nazir, Ettikan Kandasamy Karuppiah, Kim Sim Lam, et al. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57(2):245–285, 2018.

Paul-Alexandru Chirita, Wolfgang Nejdl, and Cristian Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pp. 67–74, 2005.

Anahita Davoudi and Mainak Chatterjee. Detection of profile injection attacks in social recommender systems using outlier analysis. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2714–2719. IEEE, 2017.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Xinwei Deng, V Roshan Joseph, Agus Sudjianto, and CF Jeff Wu. Active learning through sequential design, with applications to detection of money laundering. *Journal of the American Statistical Association*, 104(487):969–981, 2009.

Mahsa Ebrahimian and Rasha Kashef. Efficient detection of shilling's attacks in collaborative filtering recommendation systems using deep learning models. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pp. 460–464. IEEE, 2020.

Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th annual computer security applications conference*, pp. 381–392, 2018.

Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pp. 3019–3025, 2020.

Zengan Gao. Application of cluster-based local outlier factor algorithm in anti-money laundering. In *2009 International Conference on Management and Service Science*, pp. 1–4. IEEE, 2009.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Famei He, Xuren Wang, and Baoxu Liu. Attack detection by rough set theory in recommendation system. In *2010 IEEE International Conference on Granular Computing*, pp. 692–695. IEEE, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Jullum, Anders Løland, Ragnar Bang Huseby, Geir Ånonsen, and Johannes Lorentzen. Detecting money laundering transactions with machine learning. *Journal of Money Laundering Control*, 2020.

Shyong K Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pp. 393–402, 2004.

Karel Lannoo and Richard Parlour. Anti-money laundering in the eu: Time to get serious. ceps task force report 28 jan 2021., January 2021. URL `http://aei.pitt.edu/103318/`.

Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Chen Lin, Si Chen, Meifang Zeng, Sheng Zhang, Min Gao, and Hui Li. Shilling black-box recommender systems by learning to generate fake user profiles. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Joana Lorenz, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity. *arXiv preprint arXiv:2005.14635*, 2020.

Michael Luca. Reviews, reputation, and revenue: The case of yelp.com. *American Economic Journal - Applied Economics*, 2016.

Lin-Tao Lv, Na Ji, and Jiu-Long Zhang. A rbf neural network model for anti-money laundering. In *2008 International Conference on Wavelet Analysis and Pattern Recognition*, volume 1, pp. 209–215. IEEE, 2008.

Yelyzaveta Meleshko, Oleksandr Drieiev, and Hanna Drieieva. Method of identification bot profiles based on neural networks in recommendation systems. 2020.

Catarina Oliveira, João Torres, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. Guiltywalker: Distance to illicit nodes in the bitcoin network. *arXiv preprint arXiv:2102.05373*, 2021.

Arin Ray. It and operational spending in aml-kyc: 2021 edition, December 2021. URL `https://www.celent.com/insights/428901357`.

Saleha Raza and Sajjad Haider. Suspicious activity reporting using dynamic bayesian networks. *Procedia Computer Science*, 3:987–991, 2011.

David Savage, Qingmai Wang, Pauline Chou, Xiuzhen Zhang, and Xinghuo Yu. Detection of money laundering groups using supervised learning in networks. *arXiv preprint arXiv:1608.00708*, 2016.

Reza Soltani, Uyen Trang Nguyen, Yang Yang, Mohammad Faghani, Alaa Yagoub, and Aijun An. A new algorithm for money laundering detection based on structural similarity. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 1–7. IEEE, 2016.

Xiaobing Sun, Jiabao Zhang, Qiming Zhao, Shenghua Liu, Jinglei Chen, Ruoyu Zhuang, Huawei Shen, and Xueqi Cheng. Cubeflow: Money laundering detection with coupled tensors. In *PAKDD (1)*, pp. 78–90. Springer, 2021.

Jiaxi Tang, Hongyi Wen, and Ke Wang. Revisiting adversarially learned injection attacks against recommender systems. In *Fourteenth ACM conference on recommender systems*, pp. 318–327, 2020.

Jun Tang and Jian Yin. Developing an intelligent data discriminating system of anti-money laundering based on svm. In *2005 International conference on machine learning and cybernetics*, volume 6, pp. 3453–3457. IEEE, 2005.

Shiyu Wang, Yuanqi Du, Xiaojie Guo, Bo Pan, and Liang Zhao. Controllable data generation by deep learning: A review. *arXiv preprint arXiv:2207.09542*, 2022.

Xingqi Wang and Guang Dong. Research on money laundering detection based on improved minimum spanning tree clustering and its application. In *2009 Second international symposium on knowledge acquisition and modeling*, volume 2, pp. 62–64. IEEE, 2009.

R Cory Watkins, K Michael Reynolds, Ron Demara, Michael Georgiopoulos, Avelino Gonzalez, and Ron Eaglin. Tracking dirty proceeds: exploring data mining technologies as tools to investigate money laundering. *Police Practice and Research*, 4(2):163–178, 2003.

Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076*, 2018.

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

Fan Wu, Min Gao, Junliang Yu, Zongwei Wang, Kecheng Liu, and Xu Wang. Ready for emerging threats to recommender systems? a graph convolution-based generative shilling attack. *Information Sciences*, 578:683–701, 2021.

Hui Xia, Bin Fang, Min Gao, Hui Ma, Yuanyan Tang, and Jing Wen. A novel item anomaly detection approach against shilling attacks in collaborative recommendation systems using the dynamic time interval segmentation technique. *Information Sciences*, 306:150–165, 2015.

Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

Fuzhi Zhang and Quanqiang Zhou. Hht–svm: An online method for detecting profile injection attacks in collaborative recommender systems. *Knowledge-Based Systems*, 65:96–105, 2014.

Xuxin Zhang, Jian Chen, Rui Zhang, Chen Wang, and Ling Liu. Attacking recommender systems with plausible profile. *IEEE Transactions on Information Forensics and Security*, 16:4788–4800, 2021.

A APPENDIX

A.1 DIVERSITY MEASURE

We want to ensure that attacks are diverse. Otherwise, they are easily detected.

To quantify the diversity of the generated attacks, we calculate the inception score of three distributions extracted from the generated data:

- The amount distribution;
- The count distribution, i.e., the number of transactions per account;
- The interval distribution, i.e., the time difference between consecutive transactions of a same account.

We then define the diversity score as the average of these three inception scores (Equation 4).

$$\begin{aligned}&(\mathbb{E}_{x\sim p_G}[D_{KL}(d_{amount}(x)||\mathbb{E}_{y\sim p_G}[d_{amount}(y)])]+\\&\mathbb{E}_{x\sim p_G}[D_{KL}(d_{count}(x)||\mathbb{E}_{y\sim p_G}[d_{count}(y)])]+\\&\mathbb{E}_{x\sim p_G}[D_{KL}(d_{interval}(x)||\mathbb{E}_{y\sim p_G}[d_{interval}(y)])])/3\end{aligned} \tag{4}$$

A.2 FURTHER EXPERIMENTAL DETAILS: ANTI-MONEY LAUNDERING

Index	Layer	Output shape
0	Linear(100, 400)	(400,)
1	Linear(400, 1600)	(1600,)
2	Linear(1600, 5000)	(5000,)
	Reshape	(5,10,10,10)
3	ConvTranspose1d(10,10,4,2,1)	(5,10,20,10)
4	ConvTranspose1d(10,10,4,2,1)	(5,10,40,10)
5	ConvTranspose1d(10,10,4,2,1)	(5,10,80,10)
6	ConvTranspose1d(10,10,4,2,1)	(5,10,160,10)
	Split amounts and probabilities	(5,10,160,10)
7	ConvTranspose1d(10,10,4,2,1)	(5,10,320,10)
8	Conv1d(10,1,1,1,0)	(5,10,320)
	Categorical sampling	(5,10,320)

Table 1: AML generator architecture.

Index	Layer	Output shape
0	Conv1d(1,10,6,4,1)	(5,10,80,10)
1	Conv1d(10,10,6,4,1)	(5,10,20,10)
2	Conv1d(10,5,6,4,1)	(5,10,5,5)
	Reshape	(5,2,5,25)
	Mean pooling	(1,2,1,25)
	Concatenate amounts and probabilities	(100,)
3	Linear(100,128)	(128,)
4	Linear(128,32)	(32,)
5	Linear(32,1)	(1,)

Table 2: AML discriminator architecture.

alpha	1
beta	$[10^2, 10^5]$
gamma	$[10^3, 4 \times 10^3]$
lr	$[10^{-4}, 3 \times 10^{-3}]$

Table 3: Hyperparameter search space on the AML use case.

A.3 Comparison of real and generated Anti-Money Laundering data

In Figure 10, we compare the distributions of the total amount of money flowing through the internal accounts from real data with our generated dataset; we observe that the generators consistently move more money than real accounts. By investigating the distributions of amounts per transaction and the number of transactions per account, we verify that the generators can circulate more money than real accounts because they do more transactions but keep the amounts in a similar range as the real data.

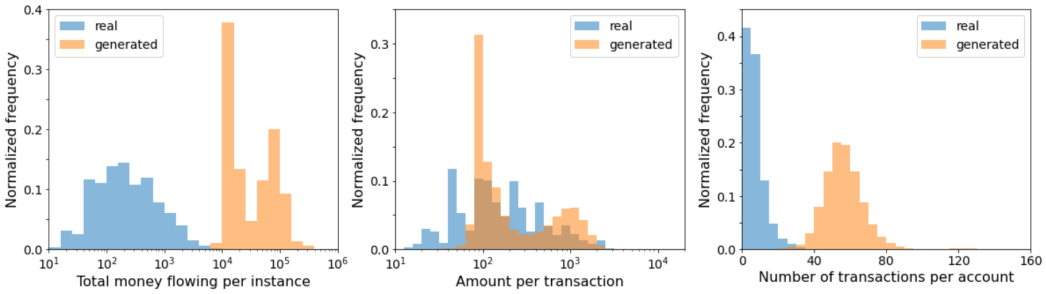


Figure 10: Comparison distributions of total money flow, amounts and counts between G and real.

A.4 Further experimental details: Recommender Systems

Index	Layer	Output shape
0	ResBlock(128,128)	(128,)
1	ResBlock(128,128)	(128,)
	Split ratings and probabilities	(128,)
2	ResBlock(128,128)	(128,)
3	ResBlock(128,64)	(64,)
4	Linear(64,3706)	(3706,)
	Categorical sampling	(3706,)

Table 4: RS generator architecture.

Index	Layer	Output shape
0	Dense(3706,64)	(64,)
1	ResBlock(64,128)	(128,)
2	ResBlock(128,64)	(64,)
	Concatenate ratings and probabilities	(128,)
3	ResBlock(128,128)	(128,)
4	ResBlock(128,128)	(128,)
5	Linear(128,1)	(1,)

Table 5: RS discriminator architecture.