Applied Intelligence
https://doi.org/10.1007/s10489-022-03378-7

# Attacking Bitcoin anonymity: generative adversarial networks for improving Bitcoin entity classification

Francesco Zola[1,2] · Lander Segurola-Gil[1] · Jan L. Bruse[1] · Mikel Galar[2] · Raul Orduna-Urrutia[1]

Accepted: 11 February 2022

## Abstract

Classification of Bitcoin entities is an important task to help Law Enforcement Agencies reduce anonymity in the Bitcoin blockchain network and to detect classes more tied to illegal activities. However, this task is strongly conditioned by a severe class imbalance in Bitcoin datasets. Existing approaches for addressing the class imbalance problem can be improved considering generative adversarial networks (GANs) that can boost data diversity. However, GANs are mainly applied in computer vision and natural language processing tasks, but not in Bitcoin entity behaviour classification where they may be useful for learning and generating synthetic behaviours. Therefore, in this work, we present a novel approach to address the class imbalance in Bitcoin entity classification by applying GANs. In particular, three GAN architectures were implemented and compared in order to find the most suitable architecture for generating Bitcoin entity behaviours. More specifically, GANs were used to address the Bitcoin imbalance problem by generating synthetic data of the less represented classes before training the final entity classifier. The results were used to evaluate the capabilities of the different GAN architectures in terms of training time, performance, repeatability, and computational costs. Finally, the results achieved by the proposed GAN-based resampling were compared with those obtained using five well-known data-level preprocessing techniques. Models trained with data resampled with our GAN-based approach achieved the highest accuracy improvements and were among the best in terms of precision, recall and f1-score. Together with Random Oversampling (ROS), GANs proved to be strong contenders in addressing Bitcoin class imbalance and consequently in reducing Bitcoin entity anonymity (overall and per-class classification performance). To the best of our knowledge, this is the first work to explore the advantages and limitations of GANs in generating specific Bitcoin data and "attacking" Bitcoin anonymity. The proposed methods ultimately demonstrate that in Bitcoin applications, GANs are indeed able to learn the data distribution and generate new samples starting from a very limited class representation, which leads to better detection of classes related to illegal activities.

**Keywords** Bitcoin address classification · Entity anonymity attack · Entity classification · Generative adversarial networks (GAN) · Class imbalance problem

## 1 Introduction

Bitcoin represents a pseudo-anonymous peer-to-peer network, which allows its users to communicate through transactions stored in a public ledger called blockchain [1]. To date, Bitcoin is the most frequently used cryptocurrency for concealing illicit activities as quantified in traffic of around $76 billion per year [2]. Users typically feel shielded by Bitcoin anonymity and at the same time conceive it as a convenient payment mechanism [3]. Furthermore, although the volatility of cryptomarkets introduces high risks, entrepreneurs and corporate executives have high expectations regarding blockchain technology. Several recommendations for blockchain platforms can be found in [4]. An escalation of illegal activities and the goal of improving the network's resilience to cyber-attacks have led researchers and Law Enforcement Agencies (LEAs) to investigate how to reduce anonymity within the Bitcoin blockchain network [5]. Anonymity can be decreased through Bitcoin entity classification [6, 7], which aims to detect and classify entities' behavioural patterns within the network. However, this classification problem - typically addressed via supervised machine learning approaches - strongly depends on the initial labelled Bitcoin dataset,

✉ Francesco Zola
fzola@vicomtech.org

Extended author information available on the last page of the article.

Published online: 01 April 2022

which allows the definition of singular entity behaviour. These kinds of datasets are often characterized by a non-homogeneously distributed population, which means that some entity classes present in the dataset are more populated than others, causing a severe *class imbalance problem*. Especially for supervised machine learning problems, this phenomenon dramatically affects the quality of the learning system. Classifiers trained using imbalanced datasets are usually biased in favour of the majority classes and fail to detect underrepresented ones [8].

The class imbalance problem becomes even more relevant in applications where it is hard to detect and collect new observations as is typically the case for Bitcoin-related data. The lack of data, usually related to samples potentially associated with illicit activities, negatively affects the performance of a classifier for a behavioural prediction system [9].

Traditional approaches used to address dataset imbalance problems can be clustered into four groups [8]: cost-sensitive learning, data-level preprocessing, algorithm-level approaches, and ensemble learning. Cost-sensitive learning uses cost values for penalizing misclassifications of some classes to improve their importance during the training phase [10]. Data-level approaches are based on adjusting the initial distribution to construct a balanced training dataset (resampling). Algorithm-level techniques try to modify existing algorithms to address the class imbalance problem directly inside the learning algorithm itself [11]. Ensemble learning is composed of techniques that train different classifiers with the original data and then combine their results for the final classification [8].

Cost-sensitive and data-level approaches, as well as a combination of both methods, are the most used techniques when deep learning models are trained [12]. In particular, cost-sensitive modifications of the back-propagation learning are made in order to improve the sensitivity of the minority class. This operation promotes the classification of samples in the minority class over the majority ones, as shown in several works [12, 13].

Other interesting and more complex approaches for addressing the imbalance problem are based on the implementation of generative models for enhancing the size and quality of the training data [14]. These approaches have been widely and very successfully used for image [15] and video [16] generation, since they are able to learn underlying true data distributions from limited available samples. In fact, as presented in [17], adversarial technology, and more specifically generative models, allow access and unlock hidden information in a dataset. Moreover, these models are typically used for enhancing the fairness in the original dataset to avoid bias in the classification [18].

For this reason, in this paper, we introduce and analyse a method based on adversarial learning (generative model) to tackle the Bitcoin class imbalance problem, with the ultimate goal of improving the performance of the final classifier. Motivated by their good results in other domains and their promising capabilities to learn complex data distributions, we use here Generative Adversarial Networks (GANs) [19] to create new synthetic samples balancing the class distributions in the original dataset, and then use the balanced dataset to train the classifier. In this way, we can evaluate how the additional synthetic samples help to improve the classifier and ultimately affect its ability to decrease Bitcoin entity anonymity. Since our approach is based on balancing the initial population by adding new elements (resampling), we compare our GAN-based methodology with commonly used data-level techniques. GANs are typically implemented using two neural networks competing with each other in order to improve the ability of the whole system to learn and reproduce the input distribution. These "adversarial" models are mainly used in the domain of image processing [20] - currently being one of the most promising machine learning models in tasks related to image generation [21, 22].

Despite their potential, it is impossible to find a unique GAN solution that works for every scenario, which has led to the creation of different GAN architectures, each one with different goals and different application domains [23]. Therefore, one aim of this study is to investigate which type of GAN architecture can be used to generate synthetic Bitcoin address data behaviour and which architecture generates the most "valuable" information, i.e. valid synthetic samples that actually improve the Bitcoin entity classification. We also investigate how GAN training time affects classification results and whether results achieved with the best performing GAN setup are consistent when repeating the experiment. Finally, 5 state-of-the-art resampling techniques are used here and compared to our GAN approach in terms of classification performance and computational efforts.

To the best of our knowledge, this is the first work exploring the benefits and limitations of GANs in generating specific Bitcoin data and investigating in detail how the generated synthetic information affects Bitcoin entity classification.

The rest of the paper is organized as follows. In Section 2, concepts regarding the class imbalance problem and GANs as well as related work are introduced. In Section 3, the proposed solutions are presented, while Section 4 details the used data, the metrics and the experiments carried out in this study. In Section 5, results are reported and discussed and finally, Section 6, provides conclusions and guidelines for future work.

## 2 Preliminaries

In this Section, we recall the concepts behind Bitcoin classification and attacking its anonymity, resampling techniques to tackle class imbalance, and GANs. In particular, in Section 2.1, the Bitcoin entity classification for attacking Bitcoin anonymity is introduced. In Section 2.2, GAN architectures and their applications and limitations are described, finally, in Section 2.3, the most frequently used techniques to address class imbalance problems and how they are applied to Bitcoin data are reported.

### 2.1 Attack on Bitcoin anonymity

Bitcoin is the dominant cryptocurrency used in criminal activities as it allows non-transparent transactions and lacks effective regulatory mechanisms [24]. Over the years, the volume of transactions, cyber-attacks on specific Bitcoin entities [25] and illicit activities related to money laundering (ransomware, Ponzi schemes etc.) have increased [26], thereby promoting an increase in the usage of services that offer to protect user anonymity (like mixers [27]). Therefore, reducing anonymity within the network and classifying Bitcoin entities have become challenging and crucial tasks for Law Enforcement Agencies (LEAs) [5].

An entity is an actor in the Bitcoin network that controls or can control multiple public keys (i.e. Bitcoin addresses) and that does not always correspond to a single physical user (organization, corporation, small group of people). Entity classification represents an attack on Bitcoin's anonymity [28] as it allows the detection of entities that have a high risk of being involved in illicit transactions [29], as well as entities that are potentially more vulnerable to cyber-attacks. In fact, as described in [30], there are classes (like markets, ransomware, mixing, etc.) composed of entities more prone to making illicit transactions, while others are composed of entities that are more prone to be attacked (like exchanges, pool, etc.) because they manage a large amount of money [31, 32].

In Bitcoin entity detection, the idea is to exploit the information available in the blockchain, i.e. blocks and transactions, in order to define entity behaviours (or classes) and thereby classify entities [6].

These transactions can be used to extract valuable information linking input and output Bitcoin addresses, as well as other characteristics such as amount, fees, times etc. (Fig. 1). The available information provides a starting point for analysing the money flow but could be insufficient for defining the different entities. In this case, heuristics or external datasets are used. In the first case, addresses are clustered into entities following assumptions that represent common behaviours in the network [33], whereas in the second case, entity characterization is based on using external private or public Bitcoin "ground-truth" datasets [30, 34], which contain clusters of addresses belonging to known entities, the name of these entities and their related classes (Exchange, Gambling, Market, etc.).

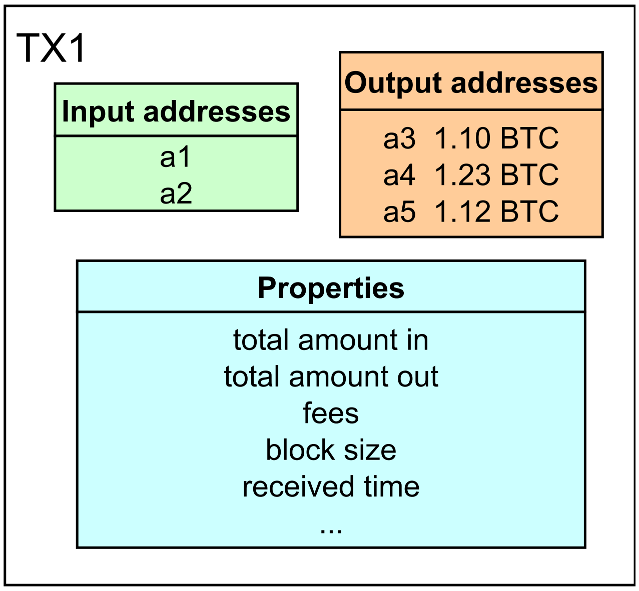


**Fig. 1** Example of information that can be extracted from a Bitcoin transaction (TX1)

Combining Bitcoin transaction data with heuristics or external datasets allows one to obtain an *address-graph* in which each address is connected to a transaction and to other addresses that belong to the same entity, as shown in Fig. 2.

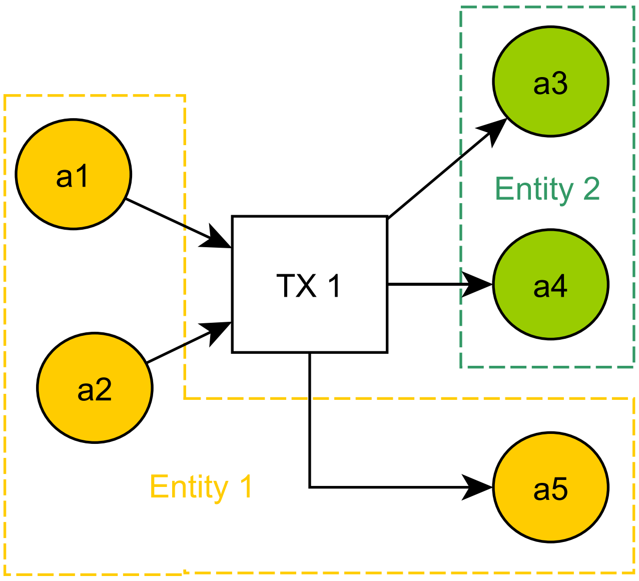


**Fig. 2** Example of address-graph extracted using the Bitcoin transaction (TX1) data

The address-graph is the starting point for many Bitcoin entity classification studies [6, 29, 34, 35].

Entity behaviours are strictly related to the addresses used to define them, so when the classification is performed at the level of address information only (using input address features only), the operation is also called Bitcoin address classification [36]. Its aim is to define and predict entity behaviour (or class) associated with one or more addresses.

First classification approaches used statistical analyses and graph information as, for example, in [37], while recent works tend to exploit the power of machine learning techniques [29, 30]. This new trend makes it appealing to transfer technologies that have already shown good results in other domains to the Bitcoin blockchain domain. However, Bitcoin datasets typically do not contain homogeneous information regarding all Bitcoin entity classes (such as Exchanges, Mixers, Mining Pools, etc.), which means that there are more data pertaining to certain classes whereas others are underrepresented. This class imbalance problem represents a major obstacle during the training phase of machine learning models [8], since it dramatically affects the quality of classification results.

## 2.2 Generative adversarial networks (GANs)

A GAN is a generative model based on the joint optimization of two neural networks. Both networks represent players in a theoretical game designed to discover, learn, and replicate input distributions. The two neural networks are called Generator (G) and Discriminator (D), according to their tasks during the training phase. The objective of G is to learn the input distribution and generate synthetic samples similar to the real ones. The objective of D is to learn the difference between the synthetic and the real data evaluating the quality of G's samples. This competition drives both networks to improve their ability to learn from each other - creating a dynamic evolution of their neural parameters. The adversarial training ends when the optimization process stops, i.e. when the synthetically generated samples are indistinguishable from the real ones [19]. The first GAN, introduced in [19], follows the architecture presented in Fig. 3.

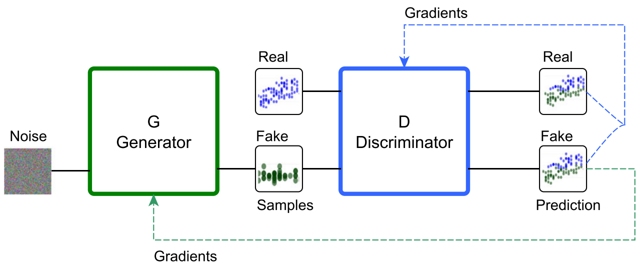


**Fig. 3** General Generative Adversarial Network (GAN) architecture

Adversarial learning is characterized by a zero-sum non-cooperative game, also called *minimax* problem. The minimax function used in GAN implementations can be formulated with the parameterized networks G and D as introduced in [19] and reported in (1). $V(D, G)$ represents the value function in the two-player game, $D(x)$ is the discriminator's estimation of the probability that $x$ (real data) is real, $E_{x \sim p_{data}(x)}$ is the expected value over all real data associated to the probability distribution of the real data $p_{data}(x)$, $G(z)$ is the generator's output (fake data) with noise input $z$, $E_{z \sim p_z(z)}$ is the expected value over all random inputs to the generator associated to a predefined prior noise distribution $p_z(z)$, and $D(G(z))$ is the discriminator's estimation of the probability that a fake sample is real.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log(D(x))] + E_{z \sim p_z(z)}[\log(1 - D(G(z))] \quad (1)$$

Following (1), one network tries to maximize the effects of its actions during the training phase, while the second network tries to minimize its effects. Since the two networks are related via a common equation, improvements of one model worsen the other one, thus creating a dynamic learning system. As introduced by Goodfellow et al. [19], the goal of the training is to find a point of equilibrium between the two competing concerns, i.e. training converges when G and D reach the well-known Nash equilibrium [38]. A Nash equilibrium occurs when one player will not change its action regardless of what the opponent may do.

In order to find the equilibrium in the cost function, gradient descent optimization is used. This optimization updates both G and D simultaneously through stochastic gradient updates [39]. However, each scenario has its own suitable optimization function and there is no unique GAN architecture that works in every situation

### GAN Limitations

Despite its learning abilities, the GAN architecture introduced in [19] comes with some problems that limit its usage compared to other architectures. Common collateral-effects [40], known as *non-convergence, vanishing gradients* and *mode collapse*, need to be taken into account at the time of implementation.

- *Non-convergence*: commonly occurs during the gradient descent optimization, where local minima and saddle points can stall the training, for example. As mentioned before, the goal of the training is to find an equilibrium in a game between two players, the generator G and the discriminator D. In a scenario where users "undo" each other's progress, no convergent solution may be reached, in which case oscillation in the models' parameters generate instability [41].

- *Vanishing gradient*: is a known problem affecting deep learning models that use gradient-based optimization. This phenomenon can produce slow training - for example when the solution follows a "pathological" curvature, which may even lead to non-convergence [42]. As presented in [43], the vanishing gradient can be produced when D is too good so that it does not provide enough information for G to learn the input distribution.
- *Mode Collapse*: also known as the Helvetica scenario, is produced when G learns to cheat D by generating only a limited variety of data regardless of the input. In this case, G learns just a small part of the input distribution that does not represent the entire population. In the worst case, the model "collapses", always generating the same sample. As presented in [41], mode collapse does not seem to be associated with any particular cost function.

Generally, there is no perfect solution to address all presented problems at once. However, previous studies tried to evaluate and mitigate some of these issues by using different GAN architectures, as proposed in Goodfellow et al. [41], for example, in which the authors showed that a modified minimax loss can be used to deal with vanishing gradients. Arjovsky et al. [44] preferred to use a Wasserstein distance as a loss function, creating the so-called Wasserstein GAN (or WGAN). The Wasserstein distance measures the distance between two probability distributions, in this case, the distribution of the data observed in the training dataset and the distribution belonging to the synthetic dataset. In this implementation, even when the supports of two distributions are located in two disjoint lower-dimensional manifolds, a smooth representation of the distance in-between is provided, which results in a better stabilization of the learning process using gradient descents. In fact, the WGAN presented in [44], showed vanishing gradient and mode collapse effects being drastically reduced. Yet, depending on the domain, WGANs can suffer from unstable training, slow convergence after weight clipping, and vanishing gradients. Local stability for both the WGAN and the traditional GAN can be guaranteed using an additional term during the gradient descent updates [45].

Another interesting architecture is based on unrolled GANs. In [46], it was shown how this technique solved the problem related to mode collapse and how it stabilized the training of GANs. In *unrolled GANs*, G not only considers the current discriminator information but also $k$ future outputs of the discriminator versions in order to discourage G to exploit local optima. In particular, G will try to take steps that D will find difficult to respond to. For $k$ steps, back-propagation occurs only to update a version of D's parameters (G's parameters being fixed) in order to allow D to optimize its performance, playing always against a specific G. The optimization is always performed through a gradient descent operation. Once the $k$ steps are done, G's parameters are updated by back-propagating through all $k$ steps ("unrolled" learning process).

In this paper, as in many other publications [47, 48], the GAN architecture introduced in [19] is called *Vanilla* GAN. Two more GAN architectures (Wasserstein [44] and unrolled GAN [46]) are presented and implemented in the following Sections.

### GAN for Cybersecurity

The main goal of these adversarial networks is to approximate the real data distribution and generate synthetic samples for enhancing/enriching the original dataset [49] - for example for creating infrared high-resolution images [50, 51], realistic vehicle images [52], for enhancing underwater images [53], for image inpainting [54] and for palmprint recognition [55]. In the speech domain, GANs are used to compute an enhancement operation [56], to improve Neural Machine Translation (NMT) results generating human-like translations [57] and for emotion recognition by creating synthetic audios from audio-visual datasets [58].

Only a few recent studies apply GANs in the context of cybersecurity. In [59] and [60], GANs are used to generate new cyber-attack samples from existing data. In the former, the goal is to balance the initial dataset and improve intrusion detection systems; in [60] the objective is to train a binary classifier (attack, no-attack) and show the benefits in terms of accuracy and f1-score generated by the balanced dataset. Mukhtar et al. [61] propose an approach based on GANs and Siamese networks for generating synthetic data of side-channel attacks. In [62], a new GAN-based framework is introduced to address the class imbalance in an encrypted traffic dataset and is compared to models trained with balanced datasets using SMOTE, ROS, and Vanilla GAN techniques.

GAN architecture and its parameters always need to be chosen carefully, depending on the given scenario. For this reason, in this work, a wide variety of optimization heuristics and GAN architectures have been compared with the aim to detect the architecture that generates "highly-valuable" synthetic samples. The "value" of synthetic samples is evaluated by analysing the improvements they generate in the classification results.

## 2.3 The class imbalance problem

In machine learning, the information available for describing a problem at hand is a key factor and the generation of a machine learning model is strongly related to the number of observations. One or more categories (or classes) in the initial dataset having more samples than others may significantly affect the training phase, generating phenomena called the class imbalance problem [63]. In this situation, supervised machine learning systems tend to be "overwhelmed" by the majority class, making it hard to discover robust patterns for under-represented classes.

As we have already mentioned, the common algorithms used for addressing class imbalance can be grouped into 4 categories. Among them, algorithm-level and cost-sensitive methods are usually more dependent on the problem and ad-hoc solutions could be required. In neural networks, considering costs may be straightforward [12] and may yield similar results as data-level techniques (e.g. random oversampling) if weights are assigned for balancing the importance of the classes. Otherwise, data-level techniques tend to be more versatile, since they are independent of the classifier used, acting directly on data. In this sense, they provide more diversity in the samples, since new synthetic data may be generated (e.g. SMOTE or ADASYN) favoring the learning in neural networks.

In this work, we are interested in analysing *data-level techniques*, which are among the most used strategies [64]. These approaches are based on resampling and thus allow us to compare their results with our GAN-based approach. In particular, data-level techniques are categorized into *over-sampling* or *under-sampling*. In the first case, data are added to the less populated classes in order to reach the same (or similar) number of elements as the majority class. In the second one, data are removed from the majority classes in order to reduce the number of samples down to the same (or similar) amount of elements that describe the least represented class. Both sampling strategies can be combined creating hybrid methods.

The usage of these strategies mitigates the problem related to unbalanced data but can produce downside effects in the supervised machine learning model [8]. For example, in under-sampling, the simplest technique (Random Under Sampling or RUS) involves removing random records from the majority class, which can cause a loss of information. The simplest implementation of over-sampling (Random Over Sampling or ROS) duplicates random observations from the minority class, which, in turn, can cause overfitting. These problems are addressed by implementations of other - more complex - resampling techniques such as Tomek Links (TL) [65], Synthetic Minority Over-sampling Techniques (SMOTE) [66] and Adaptive Synthetic (ADASYN) [67] approaches.

TL is based on a heuristic approach that uses an enhancement of the Nearest-Neighbor rule. Even though it is considered to be an under-sampling strategy (as it removes Tomek links), its result is not a uniform dataset with the same number of elements for each class.

SMOTE is an over-sampling technique that creates new instances between minority class samples that are close to each other in the feature space [66]. Typically, the algorithm first computes the $k$-nearest neighbors for a sample of the minority class and then creates the synthetic instance by randomly choosing one point in the line segment that connects one of the $k$ neighbors with the considered sample. The process is repeated until a degree of balance is reached. The number of points generated for each data sample is uniform, uniquely based on the chosen $k$.

ADASYN was implemented with the aim of improving the results obtained via the SMOTE strategy [67]. The main difference between them is how to decide the number of synthetic samples generated for a particular point: in ADASYN distribution is computed for each point, whereas in SMOTE a uniform distribution is used.

More complex approaches for addressing the imbalance problem are based on generative learning, which can be categorized into two groups [14]: traditional generative models and deep generative models. The first ones are based on traditional machine learning algorithms that usually use probability/density functions for approximating the input distribution, while the second models are based on deep learning algorithms, which allow them to learn and generate more complex distributions. For this reason, deep learning models represent more challenging and interesting structures. Well-known examples are the deep Boltzmann machine (DBM), deep belief networks (DBN), variational autoencoder (VAE) among others. More specifically, DBM and DBN are energy-based models, which means that the joint probability is defined using an energy function. In particular, for both approaches, their components are trained separately and then joined in a separate phase [68]. In this sense, the resulting generative model is more difficult to stack without causing a quality loss. On the other hand, VAE and GAN models are easier to be trained. However, although VAEs facilitate the comparison among different implementations, their performances are strongly conditioned by the reconstruction error, which can generate "blurred" samples. On the other hand, GANs allow one to learn more complex distributions and efficiently generate more realistic samples, even though they have some limitations as described in Section 2.2. It is to be noted that generative models not only increase the representation of minority classes, but by learning the entire data distribution they are able to generate more variable data. This broad generation mitigates effects like underfitting and overfitting, which are challenging problems for traditional resampling techniques like RUS and ROS. Furthermore, for other resampling techniques like SMOTE or ADASYN, synthetic samples are based on local information (neighbors) only, without taking into account the overall distribution [69]. For this reason, GAN sample generation can be seen as an “intelligent over-sampling” in which complex and high dimensional behaviours are created by analyzing all available and not just partial information.

Inspired by the promising results that GANs have demonstrated in computer vision and natural language processing tasks [50, 58], in this work, we present an approach that uses such deep generative models to address the class imbalance problem related to Bitcoin entity behaviour classification. More specifically, the ability of GANs to learn and generate data by analysing the overall data distribution and not only local information, could have a strong impact on scenarios in which entities evolve over time as it is the case in the Bitcoin domain. In these scenarios, reduced local information may be related to old and no longer relevant entity behaviour. Furthermore, learning from the overall distribution allows GANs to generalize Bitcoin entity behaviours. This generalization is useful not only for exploring more deeply the feature space and for increasing classifiers’ abilities, but could be used by the community and LEAs for investigating novel entity patterns that have not yet been detected in the Bitcoin mainnet.

Class imbalance strongly affects the quality of classification, and it represents a relevant problem associated with datasets such as the Bitcoin blockchain. In particular, in [70], the class imbalance problem is not considered, and several supervised machine learning models are implemented with the aim to identify 16 licit-illicit categories of users. Authors conclude their work highlighting that due to limited instances of classes in the training data several legal and illegal classes were misclassified. Two methods frequently used in fraud detection to tackle dataset imbalance, cost-sensitive [10] and sampling-based [11] approaches, are leveraged in [71] to address the Bitcoin class imbalance problem while implementing a machine learning model for detecting Bitcoin Ponzi schemes. In [72] and in [73], authors implement models able to detect if an address belongs to an Exchange (binary classification), resolving the class imbalance problem using RUS. In [72], a sampling technique is used over the transaction-directed hypergraph, while in [73] a sampling technique is used for removing some nodes and for guaranteeing the balance between elements of two considered classes (Exchange and not-Exchange). In [36], a model for analysing Bitcoin addresses aiming to detect abnormal activities is built, and the class imbalance problem is managed by using stratified random sampling. Harlev et al. [7] applied SMOTE in order to enrich the original dataset and train a supervised machine learning model for predicting entity classes. The model trained with the SMOTE-resampled dataset showed slight improvements in terms of overall accuracy and f1-score. These improvements were limited as although SMOTE generated improvements in f1-score values for the initial minority classes, at the same time, it decreased the values of other considered classes. Monamo et al. [74] use unsupervised machine learning (*k*-means) for detecting Bitcoin frauds. They implement z-scores, chord distance, and Hellinger distance with bagging and boosted classifiers in order to describe the input dataset and compare the results with a SMOTE-resampled dataset. The latter implementation outperformed all the considered methods generating improvements for minority classes in terms of sensitivity. Nevertheless, according to the authors, this oversampling technique decreased the reliability and increased the number of false positives.

Recently, GANs have been used for forecasting Bitcoin market prices [75] or for analysing Bitcoin price trends and the stock market [76]. In [77], we study the best parameters to be used for training adversarial algorithms for learning Bitcoin data. However, generation was limited to one specific class only using a Vanilla-GAN implementation, and without evaluating them for improving the classification task. Han et al. [78] used conditional Wasserstein GAN (WCGAN), Deep Regret Analytic (DRAGAN), and SMOTE technologies for addressing Bitcoin imbalance to perform a binary classification, i.e. distinguish legitimate and illegitimate transactions. However, they used a limited dataset gathered from a single marketplace (named Silkroad) closed in November 2014, composed of 16 distinct features. The authors acknowledged that the limited dataset and the fact that its distribution is clearly concentrated on certain values could promote the Mode Collapse effect. Furthermore, despite testing different GAN architectures, they did not achieve clear improvements in the (binary) classification task.

Inspired by the results presented in [77, 78], in this work, we present a detailed analysis of three different GAN architectures and their adaptation for addressing the Bitcoin class imbalance problem. Then, we evaluate how the generated synthetic samples improve the Bitcoin entity classification (multi-class problem). Finally, the best results obtained with GAN-resampled datasets, are compared with other state-of-the-art resampling strategies, in order to evaluate the benefits and limitations of the proposed methods.

## 3 Proposal and contributions

In this work, we present a solution based on GAN architectures for addressing the Bitcoin class imbalance problem to improve entity classification and consequently decrease Bitcoin entity anonymity. The idea is to use the adversarial technique to model the real data distribution and then perform data augmentation through synthetic data generation (over-sampling strategy). As introduced in

Section 2.3, these generative models allow to obtain a more variable dataset, which not only avoids traditional downside effects (under-sampling and over-sampling), but also helps in the generalization of the entity behaviour.

In particular, our idea is that for each under-represented (minority) class, i.e. for all the Bitcoin classes excluding the most represented class, an adversarial model is trained. This training is performed by considering each class separately. Then, each GAN is used to generate sufficient synthetic data so that each class reaches the population size of the majority class. This operation creates an enriched hybrid dataset (formed of synthetic and real data) used to train the Bitcoin classifier, which is finally tested in order to evaluate how the synthetic samples have affected entity anonymity.

In this study, we implement three different GAN architectures: the Vanilla GAN, the Wasserstein GAN (WGAN) and the unrolled GAN. In particular, the first architecture helps us validate how possible GAN downside effects affect the training of a simple GAN when Bitcoin behavioural data are used. The second architecture (WGAN) is chosen for its ability to improve the model stability and make the training process easier [44]. Finally, the third architecture (unrolled) is chosen as it allows one to improve the GAN dynamics by bringing the discriminator closer to an optimal response [46]. We think that these three chosen GAN architectures represent a good benchmark set for evaluating the benefits and limitations of such technology when applied to the Bitcoin domain.

These GAN architectures are studied and implemented in order to determine the most suitable for the Bitcoin domain, which can be used for creating an enriched version of the initial dataset.

Each architecture is evaluated in several checkpoints, i.e. using different GAN training times in terms of epochs. This operation allows us to evaluate how the generation of synthetic samples is related to GAN training epochs, and when they are affected by GAN downside effects mentioned above (overfitting). In each checkpoint, the respective data generated by GAN Generators are used to train a distinct version of the address classifier.

From this point onward, we indicate an address classifier that has been generated based on data enriched with synthetic samples produced by a GAN trained up to the $i-th$ checkpoint, as $\textit{GAN-classifier}_i$.

We repeat the implementation of the best performing GAN architecture 5 times. This approach helps us to validate the mean and standard deviation of the classifier's metrics in order to check its repeatability and robustness when a GAN-resampled dataset is used. Finally, the best performing GAN-based classifier is compared with classifiers using data generated with other state-of-the-art resampling strategies.

Our main contributions can be summarized as follow:

1. Adapting state-of-the-art GAN technologies in order to learn Bitcoin behaviours;
2. Evaluating how synthetic Bitcoin samples generated by GANs improve Bitcoin entity classification;
3. Comparing three different GAN architectures in terms of generated classification improvements;
4. Studying how training time affects the GAN learning process, the synthetic data generation and the final Bitcoin entity classification;
5. Analysing the repeatability of the best GAN approach, repeating the execution 5 times to evaluate metrics' means and standard deviations;
6. Comparing the GAN-based approach with 5 traditional data-level techniques in terms of obtained Bitcoin classification performance and computational costs.

## 4 Experimental framework

In this Section, we will first provide an overview of the data used in this study (Section 4.1), then, in Section 4.2, preprocessing steps are presented and in Section 4.3 GAN configurations are explained. In Section 4.4, the metrics used for evaluating the models are listed and in Section 4.5, the idea behind each experiment is described.

### 4.1 Dataset

For this study, we used information obtained from the Bitcoin mainnet and the WalletExplorer[1]. The whole Bitcoin blockchain was downloaded by using the Bitcoin Core[2], and in particular all blocks and transactions from the beginning until block number 570,000 were downloaded, corresponding to blocks mined until April 3rd 2019, 09:20:08 AM.

At the same time, we downloaded a labelled address-entity dataset available on the website WalletExplorer. This platform contains information about transactions, addresses and real-world entity names detected over the years. Their databases are continuously updated and thus have been used as "ground truth" for many Bitcoin-related studies, as in [29, 79]. In this study, we considered six entity classes:

- *Exchange*: entities that allow their customers to trade among cryptocurrencies or to change cryptos for fiat currencies (or vice-versa);
- *Gambling*: entities that offer gambling services based on Bitcoin currency (casino, betting, roulette, etc.);

[1]https://www.walletexplorer.com/

[2]https://bitcoin.org/en/download

- *Mining Pool*: entities composed of a group of miners that work together sharing their resources in order to reduce the volatility of their returns;
- *Mixer*: entities that offer a service to obscure the traceability of their clients' transactions;
- *Marketplace*: entities allowing to buy any kind of goods or services using cryptocurrencies. Some of them potentially related to illicit activities;
- *Service*: entities that allow users to lend Bitcoins and passively earn interests, or allow them to request a loan.

As shown in Table 1, 327 different entities and more than 16,000,000 addresses were downloaded from WalletExplorer. Exchanges represent the most populated class - its samples represent more than 44% of the entire entity dataset and about 62% of the address dataset. On the other hand, Mining Pool is the least populated class in terms of addresses with a ratio of 0.53% and the Market class is the minority in terms of entities (about 6%). This overview highlights the class imbalance problem associated with Bitcoin datasets.

Bitcoin blockchain and WalletExplorer data were combined in order to obtain a labelled address dataset, fundamental for a supervised machine learning task. This new dataset was used as a starting point for extracting the features used to define address behaviours and subsequently to create the address dataframe, as shown in Fig. 4. The address dataframe was created following concepts presented in [29], and extracting 7 features related to known Bitcoin addresses: the number of transactions in which a certain address is detected as receiver/sender, the amount of Bitcoin (BTC) received/sent from/to this address, the balance, uniqueness (if this address is used in one transaction only) and siblings.

As shown in Fig. 4, the address dataframe was spilt into training and testing datasets with a proportion of 50/50 keeping class distributions unchanged (stratified).

## 4.2 Data preprocessing

Analysing the training dataset and the dependence among its available features, it was possible to perform a reduction

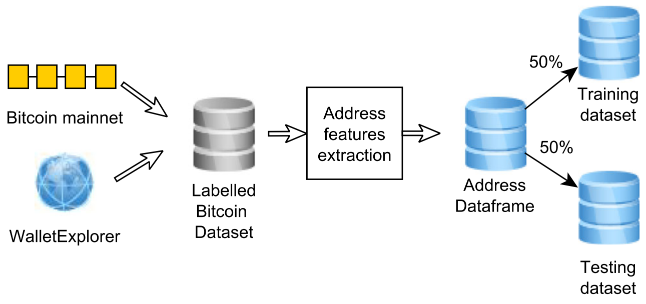


**Fig. 4** Address dataset creation

of the dataset dimensionality. The total amount of BTC received, the total amount of BTC sent and the balance are non-independent variables, so we decided to train the GAN such that it only learns two of them, whereas the third one was calculated. Thereby, the GAN learned the distribution of the amount of BTC received and the balance. The amount of BTC sent was computed a posteriori. In the same way, the uniqueness and the total received transactions are related, since one address is unique (1) when it is used exactly once for receiving money, otherwise, it is not unique (0). Following this rule, the total of received transactions was used to train the GAN, and the uniqueness values were computed a posteriori. In this manner, the 7 initial features were reduced to 5.

The training dataset was split separating the samples of each class, obtaining 6 different datasets. Each class dataset was then used for training a distinct GAN, except for the (highly populated) Exchange dataset (Fig. 5). Before training, each class dataset was normalized to reduce GAN learning complexity by limiting the feature distributions in a fixed range between 0 and 1. Let $F^m$ be the set of features that characterize each class dataset, $\forall f \in F^m$, i.e. for each specific feature, the normalization was performed following (2), where $x$ are elements in $f$, $X_{max} = \max(f)$, $X_{min} = \min(f)$, and $\tilde{x}$ represents the normalized value. Once the GANs were trained and synthetic samples could be generated, they were de-normalized using the (3) and, after computing the non-independent features, they were added to the initial (real) data, creating an Hybrid dataset, as shown in Fig. 6. The de-normalization was performed

**Table 1** Overview of the used WalletExplorer (WE) data (the overall values are in bold)

| Class | # Entity WE | % Entity WE | # Address WE | % Address WE |
|---|---|---|---|---|
| *Exchange* | 144 | 44.04 | 9,947,450 | 61.56 |
| *Gambling* | 76 | 23.24 | 3,050,899 | 18.88 |
| *Marketplace* | 20 | 6.12 | 2,349,111 | 14.54 |
| *Mining Pool* | 27 | 8.26 | 85,887 | 0.53 |
| *Mixer* | 37 | 11.31 | 475,781 | 2.94 |
| *Service* | 23 | 7.03 | 250,788 | 1.55 |
| **Total** | **327** | **100** | **16,159,916** | **100** |

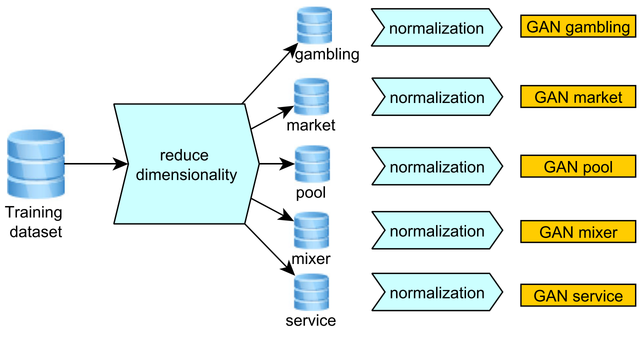


**Fig. 5** Dimensional reduction, class separation and normalization for GAN training

in order to work directly with Bitcoin data in their real space.

$$\widetilde{x} = \frac{x - X_{min}}{X_{max} - X_{min}} \qquad (2)$$

$$x = [(X_{max} - X_{min}) \times \widetilde{x}] + X_{min} \qquad (3)$$

### 4.3 GAN implementation

Despite using three different GAN architectures in this paper, the Generator (G) and the Discriminator (D) networks were the same. In [44], when working with images, the authors suggest using a 4-layer Neural Network for both G and D, where each layer is defined by 512 neurons. In our case, since the size of the Bitcoin feature space is lower than the ones used for the images, three hidden layers for both G and D were implemented, following the specifications described in [77]. In particular, G's neural network was composed of three hidden layers with 512, 256 and 128 neurons (Fig. 7), all using the Rectified Linear Unit (ReLu) as activation function [77]. The output was fixed to 5, i.e. the same number of non-independent features in the real samples. A technique called Root Mean Square Propagation (RMSProp [80]) was used in all implementations to optimize the relative cost function with a learning rate set to $5e-5$, as indicated in [44].

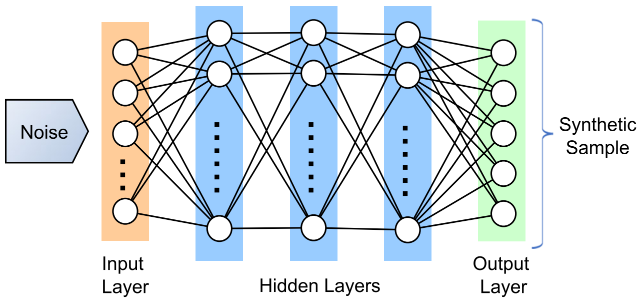


**Fig. 7** Generator architecture

The value of the batch size, representing the number of elements used at once for updating the weights of the neural networks, was kept fixed to 1,000 samples. Furthermore, for the unrolled GAN, the $k$ value of the steps forward was chosen to be equal to 5, as used in several tests in the introduced paper [46].

D's neural network was also composed of three hidden layers with 256, 512 and 256 neurons (Fig. 8), all again using the Rectified Linear Unit (ReLu) as activation function [77]. As shown in Fig. 8, the input size of the Discriminator was 5, i.e. equal to the number of non-independent real features as well as to the number of synthetically generated features.

For each architecture, 6 GAN training time checkpoints were fixed a priori. These values represent the training length of the adversarial networks, indicated in epochs. In particular, checkpoints were fixed in: 1,000, 10,000, 25,000, 50,000, 75,000 and 100,000 epochs.

### 4.4 Evaluation metrics

This Section describes the classification metrics used to evaluate and compare the different machine learning models in our experiments. Assuming $N$ to be the total number of classes and for each class $i$ assuming $tp_i$ as true positive value, $fp_i$ as false positive value, $tn_i$ as true negative value, $fn_i$ as false negative value, the following metrics were defined.

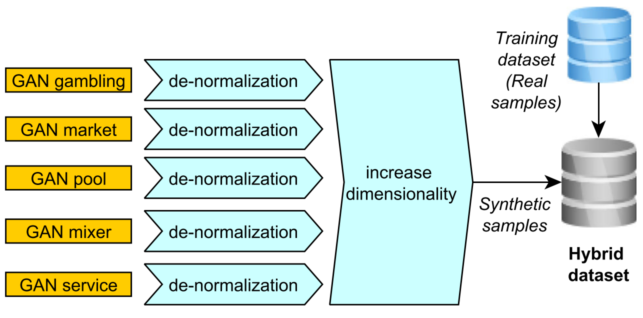


**Fig. 6** Creation of the hybrid (augmented) dataset

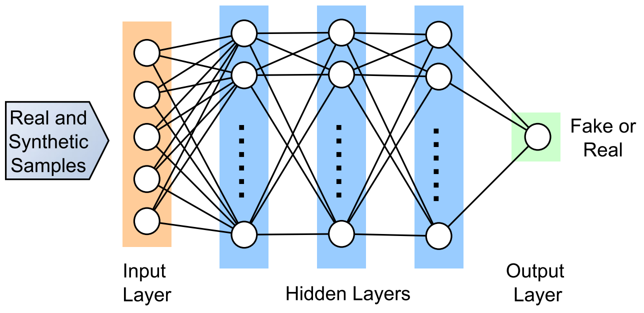


**Fig. 8** Discriminator architecture

- *Accuracy* or *Score* is defined as the number of correct predictions divided by the total number of predictions and is given as percentage (4).

$$\frac{\sum_{i=1}^{N} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{N} \tag{4}$$

- *Precision (prec.)* is the number of positive predictions divided by the total number of the positive class values predicted. It represents a measure of a classifier's exactness given as a value between 0 and 1, with 1 relating to high precision (5).

$$\frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N}(tp_i + fp_i)} \tag{5}$$

- *Recall or Sensitivity* represents a measure of a classifier's completeness quantifying the number of positive class predictions made over all positive examples in the dataset ((6)). It is given as a value between 0 and 1

$$\frac{\sum_{i=1}^{N} tp_i}{\sum_{i=1}^{N}(tp_i + fn_i)} \tag{6}$$

- *f1-score* is the harmonic mean of Precision and Recall. It takes values between 0 and 1, with 1 relating to perfect Precision and Recall and can be calculated using (7)

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

- *Matthews Correlation Coefficient (MCC)* is a metric yielding easy comparison with respect to a random baseline, particularly appropriate for unbalanced classes. It takes values between $-1$ and $+1$. A coefficient of $+1$ represents a perfect prediction, 0 an average random prediction and $-1$ an inverse prediction. As shown in [81], let $K$ be the number of classes and $C$ be a confusion matrix with dimensions $K \times K$, the $MCC$ can be calculated as shown in (10)

$$MCC_{p1} = \sqrt{\sum_k \left(\sum_l C_{kl}\right)\left(\sum_{f,g|f\neq g} C_{gf}\right)} \tag{8}$$

$$MCC_{p2} = \sqrt{\sum_k \left(\sum_l C_{lk}\right)\left(\sum_{f,g|f\neq g} C_{fg}\right)} \tag{9}$$

$$MCC = \frac{\sum_k \sum_l \sum_m C_{kk}C_{lm} - C_{kl}C_{mk}}{MCC_{p1} \times MCC_{p2}} \tag{10}$$

- *Area Under the Receiver Operating Characteristic Curve (AUC)* measures the two-dimensional area underneath the ROC curve, which is a plot of the true positive rate against the false positive rate. It indicates the classifier's ability to avoid false classification and it takes values between 0 and 1, with 1 relating to perfect predictions. AUC can be calculated for a binary classification as described in (11). This equation can be extended for multi-class problems with some adjustments.

$$AUC = \frac{1}{2}\left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp}\right) \tag{11}$$

Furthermore, the term *avg.* is used in the following to indicate average results of an indicated metric and the term *std.* is used to indicate the computed standard deviation.

## 4.5 Overview of the planned experiments

In this study, four experiments were carried out in order to explore the GAN approach for addressing the Bitcoin class imbalance problem. The obtained results were then compared with current state-of-the-art techniques in order to validate and check the limitations of the introduced methods.

The first experiment introduces the baseline classifier by directly using the training dataset obtained from the address dataframe (Fig. 9). The creation of this baseline model not only was used for detecting limitations in the Bitcoin entity classification but was also used later for evaluating improvements achieved by other presented experiments involving resampled data. The baseline model was implemented using a Random Forest model. Following our previous study on Bitcoin classification [29], the number of estimators was set to 10, and the Gini function was used to measure the quality of the split without fixing a maximum depth of the tree. The testing dataset was used to compute a first evaluation of how the baseline classifier (trained with real data) predicts entity classes related to a certain address.

In the second experiment, our solution based on adversarial learning through GANs was implemented in order to generate synthetic Bitcoin address behaviours. The experiment started by training GANs based on the training dataset. In particular, as described in Section 4.2, a single GAN was implemented for each class that was underrepresented in the training dataset (Gambling, Market, Mining Pool, Mixer, and Service). Following this specification, the GAN trained using on the largest dataset was the GAN for generating synthetic Gambling data with 1,527,247 real samples, while the GAN using the smallest dataset was the Mining Pool GAN based on 43,265 real samples only (Table 2).

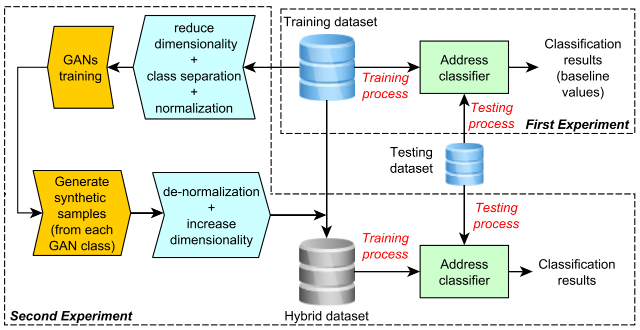


**Fig. 9** Schema of the first and second experiment

At the end of the training process, the Generators G of each class GAN were used to create new synthetic samples that were then de-normalized and joined with the real training dataset (after computing the 2 dependent features), creating the hybrid dataset (Section 4.2). Finally, this hybrid information was used to train an address classifier (Fig. 9), which was based on Random Forest models with the same setup as in the first experiment. Once this classifier was ready, it was tested using the testing dataset. This experiment was carried out by stopping the GANs' training processes in 6 different checkpoints (epochs), as explained in Section 4.3. In each one of them, the Gs were used to enrich the real dataset creating different versions of the hybrid dataset, thus creating a specific version $i$ of the GAN-classifiers (address classifiers).

**Algorithm 1** Pseudo-algorithm followed to evaluate each GAN architecture.

```
1:  Split the initial dataset per class;
2:  Normalizeeach class dataset;
3:  Reduce dimensionalityof each class dataset;
4:  for $i$ in checkpoints do
5:      Training a GANfor each class dataset;
6:      Generate synthetic samples(until each minority class reaches a population comparable to the majority one);
7:      De-normalize synthetic data and increase the dimensionality;
8:      Enrich the training datasetwith synthetic data
9:      Train GAN-classifier$_i$ with the new enriched dataset;
10:     Test GAN-classifier$_i$ with an unseen dataset(testing dataset);
11: end for
```

The steps presented in Algorithm 1 were repeated using the three types of GAN architectures, Vanilla GAN, Wasserstein GAN (WGAN), and unrolled GAN. The latter two techniques were used in order to evaluate and mitigate possible collateral effects that typically affect adversarial networks.

In the third experiment, the most promising solutions detected in the second experiment were analysed in detail. In particular, only architectures and checkpoints that had shown the highest classification values were used. For this configuration, the GAN training and the classifier training were repeated 5 more times, in order to check the repeatability of our results and avoid outlier solutions. In the fourth and last experiment, the classification results obtained in the third experiment were compared with classification results obtained via several known resampling techniques (Section 2.3). In this experiment, two under-sampling techniques, RUS and TL, and three over-sampling techniques, ROS, SMOTE, and ADASYN, were used. These resampling techniques were directly applied to the training dataset in order to use this enhanced dataset to train the Bitcoin entity classifier. The subsequent model was again a Random Forest classifier with the setup from the previous experiments and was tested using the same testing dataset.

The fourth experiment allowed us to determine the suitability of current techniques for addressing class imbalance problems in the Bitcoin domain and allowed us to find insights about how these techniques affect Bitcoin entity behaviour classification. Furthermore, comparing resampling techniques with our GAN approach highlighted the strengths and weaknesses of applying GANs to Bitcoin data.

## 5 Experimental study

In this Section, the four experiments with their obtained results are presented. In particular, in Section 5.1, the baseline model is built, in Section 5.2, the three GAN architectures are implemented and compared. Then, in Section 5.3, the best GAN solutions are used to check the repeatability of the results, whereas in Section 5.4, the GAN approach is compared with 5 resampling techniques. Finally, in Section 5.5, results are discussed.

### 5.1 Baseline model

Table 2 provides an overview of the results obtained by testing the baseline model. In particular, the number of real samples used for training the baseline model is reported as well as the values for precision, recall and f1-score per class, an overall average of these three metrics and the overall accuracy obtained using the entire testing dataset.

The baseline model showed an overall good accuracy value of 94.67%, however, it presented problems in detecting samples belonging to minority classes, which suggests

**Table 2** Baseline classifier accuracy and f1-score obtained with the testing dataset (the average values are in bold)

| Class | # train samples | prec. | recall | f1 score |
|---|---|---|---|---|
| *Exchange* | 4,972,019 | 0.95 | 0.98 | 0.96 |
| *Gambling* | 1,527,247 | 0.93 | 0.89 | 0.91 |
| *Marketplace* | 1,174,468 | 0.99 | 0.97 | 0.98 |
| *Mining Pool* | 43,265 | 0.89 | 0.75 | 0.82 |
| *Mixer* | 237,725 | 0.86 | 0.80 | 0.83 |
| *Service* | 125,531 | 0.84 | 0.67 | 0.75 |
| avg. | | **0.91** | **0.84** | **0,87** |
| score % | **94.67** | | | |

that the class imbalance problem is affecting the classification of minority classes. While majority classes were detected with high f1-score values over 0.90 (Exchange, Gambling and Marketplace), less represented classes (Mining Pool, Mixer, and Service) yielded respectively lower values of 0.82, 0.83 and 0.75.

The f1-score values were conditioned by lower recall values, indicating problems in detecting underrepresented classes. For example, among all the Service samples in the testing dataset, only 67% (i.e. recall 0.67) and only 75% and 80% of the Mining Pool and Mixer elements were correctly classified.

The obtained results underline again the importance of having a balanced dataset in multi-class classification problems using supervised machine learning. Results were affected by under-represented classes in the original Bitcoin dataset.

## 5.2 Comparison among GAN architectures

GANs were used for generating synthetic data to enrich the training dataset and create a hybrid dataset. The number of synthetic samples generated by each GANs is correlated with the class population of the initial training dataset (Table 2). The idea is to balance the initial class distributions by adding a sufficient amount of synthetic samples for each minority class. In particular, as shown in Fig. 10, the most populated class - Exchange - has 5,000,000 real samples, hence we chose to generate 3,000,000 synthetic samples for the Gambling class, 3,500,000 synthetic samples for the Market class, and 4,500,000 synthetic samples for Pool, Mixer, and Service class, respectively for each checkpoint. In this manner, the new hybrid dataset contained a new per-class distribution varied in a range between 4,527,247 (Gambling class) and 4,972,019 (Exchange class).

Once the hybrid dataset belonging to a specific checkpoint (epoch) was generated, it was denormalized and the dependent features were computed. This new dataset was used to train the GAN-classifier following the same process used for creating the baseline model. This GAN-classifier was then tested with the testing dataset, in order to evaluate its global accuracy and precision, recall, and f1-score for each class. This comparison over the same testing dataset allowed us to evaluate how much the new synthetic information generated by the GANs affected the entity classification and consequently how much the GANs have learned from the real distribution.

Results regarding the Vanilla GAN architecture are reported in Table 3; in particular the values for f1-score, precision and recall obtained from each GAN-classifier and for each class are indicated. These metrics improved with respect to the ones obtained with the baseline model, as well as the overall accuracy and the average f1-score. The two best solutions, GAN-classifier$_1$ and GAN-classifier$_3$, achieved by training the GANs with 1,000 and 25,000 epochs, in terms of overall recall reaching both values of 0.90 and in terms of f1-score as well, which was, respectively 0.05 and 0.06 points higher than the baseline values. Improvements were also visible analysing the metrics of each class individually. Even the majority class (Exchange), which was not affected by the resampling strategy, increased its f1-score by 0.02 for the two best solutions. From the results shown in Table 3, one can observe that classifiers implemented with the hybrid dataset

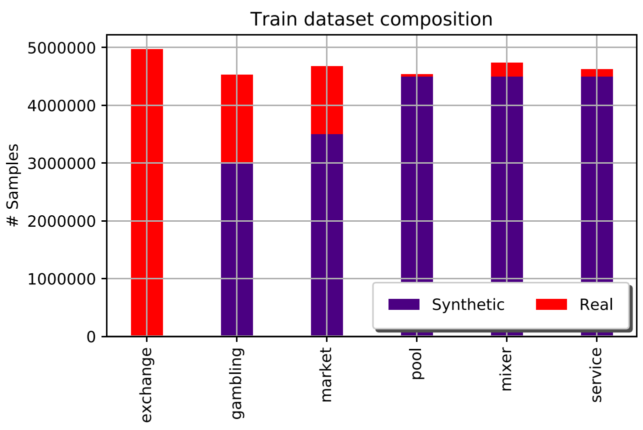


**Fig. 10** Synthetic and real sample distribution in the newly created hybrid dataset

generated by the Vanilla GANs which were trained with a low number of epochs (1,000 and 25,000) performed much better than the other ones.

The same study as presented for the Vanilla GAN was carried out by using a WGAN architecture. The obtained results are reported in Table 4. They showed a different trend, as the classifiers with the best global f1-score were obtained by creating the hybrid dataset using GANs that were trained longer, i.e. the ones trained in 50,000, 75,000 and 100,000 epochs. These three best options achieved an average f1-score of 0.90. However, the WGAN approach presented problems in detecting Mining Pool samples. In fact, although improvements in terms of recall were generated, precision values dropped, causing a decrease of f1-scores related to that class (about 0.2 - 0.3 below the baseline result).

In Table 5, computed evaluation metrics for the unrolled GAN architecture are reported. In this case, results tended to follow the trend of the Vanilla GANs, where best solutions were obtained with GANs that were trained less (in terms of epochs), yet without reaching high scores. The best solutions were obtained by using GAN-classifier$_1$ and GAN-classifier$_3$ (unrolled GAN trained for 1,000 and 25,000 epochs), showing the highest values in terms of precision, recall and f1-score.

As can be seen from Table 6, all GAN solutions show higher values in terms of AUC (0.98-0.99) regardless of the chosen architecture and checkpoint. On the other hand, both accuracy and MCC follow the trend of the averaged f1-score shown in Tables 3, 4, and 5. The Vanilla GAN shows the best accuracy and MCC for models trained respectively with 1,000 and 25,000 epochs, similar to the unrolled GAN, whereas the WGAN shows the highest accuracy and MCC values in 50,000, 75,000 and 100,000 epochs.

Comparing the average recall and f1-scores achieved with all GAN-classifiers implemented with the three proposed GAN architectures, we found that the best solutions with the highest improvements in Bitcoin entity classification were obtained using the Vanilla GAN architecture. In fact, the Vanilla GAN yielded important improvements in terms of minority class detection, as shown by high values of recall and f1-scores.

In Fig. 11, the overall accuracy of the GAN-classifiers implemented using the three GAN architectures are compared. The accuracy of the GAN-classifiers trained with data generated from the Vanilla GAN model was usually higher than the accuracy values obtained with the classifiers trained based on data generated by the other two GAN architectures. However, moving to longer epochs, the downward trend in terms of accuracy that we observed may be a symptom of mode collapse: potentially, the GAN stopped learning the real distribution and started generating similar samples only, which did not contribute

**Table 3** Precision, recall and f1-scores obtained with the GAN-classifiers at different epochs (training dataset enriched with synthetic samples generated by the Vanilla GAN). Average values are in bold

| | 1,000 epochs (GAN-classifier$_1$) | | | 10,000 epochs (GAN-classifier$_2$) | | | 25,000 epochs (GAN-classifier$_3$) | | | 50,000 epochs (GAN-classifier$_4$) | | | 75,000 epochs (GAN-classifier$_5$) | | | 100,000 epochs (GAN-classifier$_6$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score |
| Exchange | 0.97 | 0.99 | 0.98 | 0.96 | 0.98 | 0.97 | 0.96 | 0.99 | 0.98 | 0.96 | 0.99 | 0.97 | 0.96 | 0.98 | 0.97 | 0.95 | 0.98 | 0.97 |
| Gambling | 0.97 | 0.93 | 0.95 | 0.95 | 0.91 | 0.93 | 0.96 | 0.92 | 0.94 | 0.96 | 0.92 | 0.94 | 0.95 | 0.91 | 0.93 | 0.94 | 0.90 | 0.92 |
| Market | 0.99 | 0.98 | 0.99 | 0.99 | 0.97 | 0.98 | 0.99 | 0.98 | 0.98 | 0.99 | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 |
| Pool | 0.82 | 0.83 | 0.82 | 0.77 | 0.81 | 0.79 | 0.94 | 0.84 | 0.89 | 0.92 | 0.82 | 0.87 | 0.91 | 0.79 | 0.85 | 0.90 | 0.77 | 0.83 |
| Mixer | 0.91 | 0.86 | 0.88 | 0.92 | 0.85 | 0.88 | 0.92 | 0.87 | 0.90 | 0.89 | 0.83 | 0.86 | 0.88 | 0.82 | 0.85 | 0.87 | 0.81 | 0.84 |
| Service | 0.95 | 0.83 | 0.89 | 0.90 | 0.74 | 0.81 | 0.94 | 0.80 | 0.86 | 0.93 | 0.79 | 0.86 | 0.92 | 0.77 | 0.84 | 0.87 | 0.70 | 0.78 |
| avg. | **0.94** | **0.90** | **0.92** | **0.92** | **0.88** | **0.89** | **0.95** | **0.90** | **0.93** | **0.94** | **0.89** | **0.91** | **0.94** | **0.87** | **0.90** | **0.92** | **0.86** | **0.89** |

**Table 4** Precision, recall and f1-scores obtained with the GAN-classifiers at different epochs (training dataset enriched with synthetic samples generated by the WGAN). Average values are in bold

| | 1,000 epochs (GAN-classifier$_1$) | | | 10,000 epochs (GAN-classifier$_2$) | | | 25,000 epochs (GAN-classifier$_3$) | | | 50,000 epochs (GAN-classifier$_4$) | | | 75,000 epochs (GAN-classifier$_5$) | | | 100,000 epochs (GAN-classifier$_6$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score |
| Exchange | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 |
| Gambling | 0.95 | 0.91 | 0.93 | 0.94 | 0.91 | 0.93 | 0.94 | 0.90 | 0.92 | 0.95 | 0.92 | 0.94 | 0.95 | 0.91 | 0.93 | 0.95 | 0.92 | 0.93 |
| Market | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.98 | 0.98 | 0.99 | 0.98 | 0.98 | 0.99 | 0.98 | 0.98 |
| Pool | 0.86 | 0.76 | 0.81 | 0.85 | 0.76 | 0.80 | 0.80 | 0.78 | 0.79 | 0.81 | 0.78 | 0.80 | 0.79 | 0.79 | 0.79 | 0.79 | 0.80 | 0.79 |
| Mixer | 0.86 | 0.81 | 0.84 | 0.89 | 0.83 | 0.86 | 0.87 | 0.81 | 0.84 | 0.89 | 0.84 | 0.87 | 0.89 | 0.83 | 0.86 | 0.90 | 0.84 | 0.87 |
| Service | 0.87 | 0.74 | 0.80 | 0.90 | 0.74 | 0.81 | 0.83 | 0.72 | 0.77 | 0.90 | 0.78 | 0.83 | 0.89 | 0.78 | 0.83 | 0.90 | 0.76 | 0.83 |
| avg. | **0.92** | **0.86** | **0.89** | **0.92** | **0.87** | **0.89** | **0.90** | **0.86** | **0.88** | **0.92** | **0.88** | **0.90** | **0.91** | **0.88** | **0.90** | **0.92** | **0.88** | **0.90** |

**Table 5** Precision, recall and f1-scores obtained with the GAN-classifiers at different epochs (training dataset enriched with synthetic samples generated by the unrolled GAN). Average values are in bold

| | 1,000 epochs (GAN-classifier$_1$) | | | 10,000 epochs (GAN-classifier$_2$) | | | 25,000 epochs (GAN-classifier$_3$) | | | 50,000 epochs (GAN-classifier$_4$) | | | 75,000 epochs (GAN-classifier$_5$) | | | 100,000 epochs (GAN-classifier$_6$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Class | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score | prec. | recall | f1 score |
| Exchange | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.98 | 0.97 | 0.95 | 0.98 | 0.97 | 0.95 | 0.98 | 0.97 |
| Gambling | 0.95 | 0.90 | 0.93 | 0.95 | 0.90 | 0.92 | 0.95 | 0.91 | 0.93 | 0.95 | 0.90 | 0.93 | 0.94 | 0.89 | 0.92 | 0.95 | 0.90 | 0.92 |
| Market | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 |
| Pool | 0.89 | 0.78 | 0.83 | 0.93 | 0.80 | 0.86 | 0.93 | 0.80 | 0.86 | 0.91 | 0.78 | 0.84 | 0.91 | 0.78 | 0.84 | 0.91 | 0.77 | 0.84 |
| Mixer | 0.91 | 0.85 | 0.88 | 0.94 | 0.87 | 0.90 | 0.91 | 0.85 | 0.88 | 0.89 | 0.82 | 0.86 | 0.90 | 0.83 | 0.86 | 0.90 | 0.84 | 0.87 |
| Service | 0.92 | 0.77 | 0.84 | 0.88 | 0.72 | 0.79 | 0.92 | 0.76 | 0.83 | 0.92 | 0.77 | 0.83 | 0.89 | 0.71 | 0.79 | 0.89 | 0.71 | 0.79 |
| avg. | **0.94** | **0.88** | **0.90** | **0.94** | **0.87** | **0.90** | **0.94** | **0.88** | **0.91** | **0.94** | **0.87** | **0.90** | **0.93** | **0.86** | **0.89** | **0.93** | **0.86** | **0.90** |

**Table 6** Overall accuracy, MCC and AUC of each GAN-classifier at different epochs separated per architecture

| | Vanilla GAN | | | WGAN | | | unrolled GAN | | |
|---|---|---|---|---|---|---|---|---|---|
| # epochs | score % | MCC | AUC | score % | MCC | AUC | score % | MCC | AUC |
| **1,000** | 96.85 | 0.94 | 0.99 | 95.74 | 0.92 | 0.98 | 95.86 | 0.93 | 0.99 |
| **10,000** | 95.93 | 0.93 | 0.99 | 95.66 | 0.92 | 0.99 | 95.69 | 0.92 | 0.99 |
| **25,000** | 96.60 | 0.94 | 0.99 | 95.23 | 0.91 | 0.98 | 95.99 | 0.93 | 0.99 |
| **50,000** | 96.26 | 0.93 | 0.99 | 96.12 | 0.93 | 0.99 | 95.74 | 0.92 | 0.99 |
| **75,000** | 95.93 | 0.93 | 0.99 | 95.98 | 0.93 | 0.99 | 95.23 | 0.92 | 0.99 |
| **100,000** | 95.23 | 0.91 | 0.98 | 96.03 | 0.93 | 0.99 | 95.53 | 0.92 | 0.99 |

to any new knowledge ("low-valuable" samples). The two most promising configurations obtained with the Vanilla GAN, GAN-classifier$_1$ and GAN-classifier$_3$, were analysed in-depth in the third experiment.

## 5.3 Randomness of GANs

For the next experiment, GAN-classifier$_1$ and GAN-classifier$_3$ (trained for 1,000 and 25,000 epochs, respectively) were run five more times and tested. In particular, 5 new hybrid datasets were generated using 5 new generators. These new hybrid datasets were then used to train as many address classifiers (GAN-classifiers), each one tested again using the testing dataset.

In Table 7, the values of global accuracy, precision, recall and f1-scores as well as the averages of the metrics over the 5 repetitions and their standard deviations are shown for each new implementation. The best results were again obtained with the model based on the data generated from GANs trained for 1,000 epochs (GAN-classifier$_1$); its metrics showed good repeatability with 96.12% as the lowest value of accuracy and 96.96% as the greatest, with a global standard deviation of 0.282%. Slightly more unstable were the values obtained with models using data generated from GANs trained for 25,000 epochs (GAN-classifier$_3$). In this case, accuracy went down to 95.30% and up to 96.60%, with a global standard deviation of 0.472%.

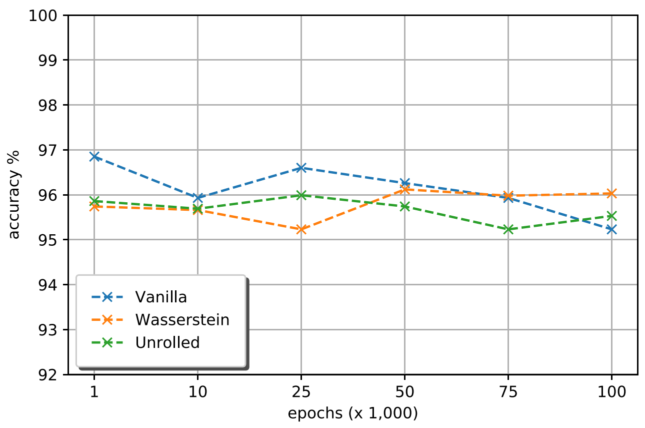


**Fig. 11** Comparison of overall accuracy values obtained with the GAN-classifiers at different epochs

Figure 12 shows a comparison of f1-scores averaged over the 5 repetitions for each entity class. Both GAN-classifier$_1$ and GAN-classifier$_3$, presented very limited variability related to Exchange, Gambling and Market classification. Furthermore, GAN-classifier$_1$ presented high repeatability (limited variance) for the other three classes, while GAN-classifier$_3$ showed higher variability related to the Pool, Mixer and Service classes.

## 5.4 GANs for resampling vs. classical resampling methods

In the fourth and last experiment, the best solutions from prior experiments were compared to several common resampling techniques introduced in Section 2.3.

Table 8 indicates the per-class sample population after applying each additional resampling technique. Techniques such as RUS remove samples in order to reach the sample size present in the minority class, and techniques like ROS, SMOTE and ADASYN tend to create new samples in order to match the majority class. Interestingly, the Tomek links strategy did not change dramatically class distribution, even though it removes unwanted overlap between classes.

In Table 9, the values of overall accuracy, precision, recall, f1-score, MCC and AUC are reported, which were obtained by using the same testing dataset over the baseline model and over Bitcoin entity classifiers, in which the class imbalance problem was addressed by applying common resampling strategies and the novel GAN-based strategy introduced in this paper. The results shown in Table 9 can be compared with the baseline results and can be interpreted by dividing them into three categories: strategies that negatively affect classification results, strategies that do not seem to affect results and strategies that generate improvements.

The RUS technique falls into the first category as, in fact, this under-sampling algorithm removed important

**Table 7** Overall accuracy, precision, recall and f1-scores obtained with GAN-classifier$_1$ and GAN-classifier$_3$ in 5 implementations

| | 1,000 (GAN-classifier$_1$) | | | | 25,000 (GAN-classifier$_3$) | | | |
|---|---|---|---|---|---|---|---|---|
| Repetition # | score % | prec. | recall | f1 score | score % | prec. | recall | f1 score |
| 1 | 96.12 | 0.93 | 0.88 | 0.90 | 96.34 | 0.94 | 0.89 | 0.91 |
| 2 | 96.96 | 0.94 | 0.90 | 0.92 | 96.55 | 0.95 | 0.90 | 0.92 |
| 3 | 96.71 | 0.93 | 0.90 | 0.91 | 95.30 | 0.92 | 0.86 | 0.89 |
| 4 | 96.78 | 0.95 | 0.91 | 0.93 | 96.17 | 0.94 | 0.89 | 0.91 |
| 5 | 96.62 | 0.93 | 0.90 | 0.92 | 96.60 | 0.95 | 0.90 | 0.93 |
| avg. | **96.64** | **0.94** | **0.90** | **0.92** | **96.19** | **0.94** | **0.89** | **0.91** |
| std. | **0.282** | **0.007** | **0.010** | **0.075** | **0.472** | **0.012** | **0.014** | **0.013** |

In each one, the classifiers were trained with different hybrid datasets (newly generated using the best Vanilla GAN configuration). Average and standard values are in bold

information and generated a quality loss in the classification - reflected in a worsening of all the considered metrics. Accuracy decreased by more than 10% and the f1-score was 0.03 points below the baseline metric.

Tomek links and ADASYN were included in the second category. In particular, the under-sampling technique did not change strongly the initial dataset, and thus its evaluation metrics matched the baseline values. Although the ADASYN strategy generated improvements, we consider them to be insignificant here as the accuracy was just 0.12% and the f1-score was only 0.01 points above the baseline values.

ROS, SMOTE and GANs strategies belonged to the third category, as these techniques generated visible improvements in the results compared to values obtained by using the imbalanced Bitcoin dataset. In terms of overall accuracy, the best solution was obtained by addressing the imbalance problem using the GAN approach introduced in this paper, and in particular using GAN-classifier$_1$ i.e. a configuration of Vanilla GANs trained for 1,000 epochs. Our strategy showed an accuracy value of 96.64%; 0.07%

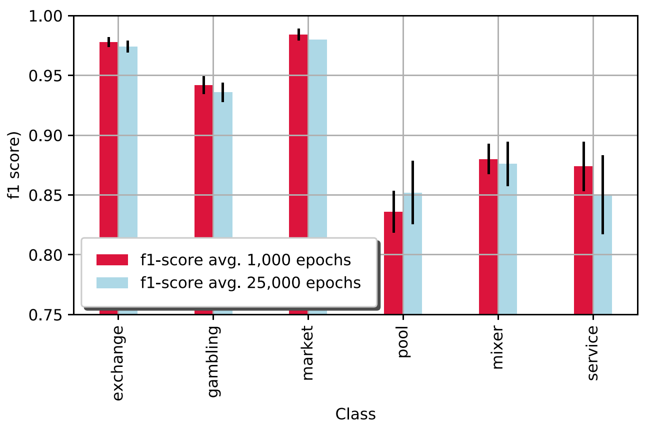


**Fig. 12** Comparison of GAN-classifier$_1$ and GAN-classifier$_3$ f1-scores averaged over the 5 repetitions and standard deviations for each entity class

above the second-best accuracy value achieved by the ROS strategy, and 1.97% above the baseline model. ROS and GAN strategies shared the same values of MCC and AUC (0.94 and 0.99, respectively). However, in terms of precision, recall and f1-score the ROS technique generated slightly better results, reaching an f1-score of 0.94 versus 0.92 obtained with the GAN strategy.

Furthermore, Table 9 shows that GAN-classifier$_3$ generated better results than the majority of the presented resampling techniques as well, in fact only GAN-classifier$_1$ and ROS performed better.

In Fig. 13, improvements and breakdowns in terms of precision are shown, and it is evident that the model trained with data enriched by our GAN strategy, together with ROS, were the best solutions. It is to be noted that the GAN strategy showed a decrease of precision regarding the Pool class (0.04 below baseline), while the model trained based on ROS data did not present such problems (0.04 above baseline). Yet, for the Gambling class, it was the GAN strategy that generated improvements (0.03 above the baseline), while the model trained with ROS data did not achieve this precision. Both techniques did not generate precision improvement for the Market class, but they generated similar improvements with respect to other classes. For example, in the Service class, a value of 0.1 points above the baseline model was obtained.

Figure 14 shows the breakdown of scores achieved per class in terms of recall. The best results were obtained using the SMOTE strategy. This strategy generated improvements in almost all classes. For the Service class, for example, the recall was 0.28 points above the baseline score; only in the Exchange class a lower score was registered (-0.03). The RUS trend was aligned with the theory of under-sampling techniques - removing random samples from the majority classes in order to balance the dataset, recall scores increased for underrepresented classes, while they were dramatically decreased for Exchange, Gambling and Market classes. Our GAN strategy improved recall values in all

**Table 8** Populations in resampled datasets using 5 different resampling strategies

| | RUS | TL | ROS | SMOTE | ADASYN |
|---|---|---|---|---|---|
| Class | Train size | Train size | Train size | Train size | Train size |
| Exchange | 43,265 | 4,967,951 | 4,972,019 | 4,972,019 | 4,972,019 |
| Gambling | 43,265 | 1,524,202 | 4,972,019 | 4,972,019 | 4,975,061 |
| Market | 43,265 | 1,173,795 | 4,972,019 | 4,972,019 | 4,971,980 |
| Pool | 43,265 | 43,265 | 4,972,019 | 4,972,019 | 4,971,821 |
| Mixer | 43,265 | 236,155 | 4,972,019 | 4,972,019 | 4,972,000 |
| Service | 43,265 | 125,019 | 4,972,019 | 4,972,019 | 4,971,911 |

classes, except for the Market class. Nevertheless, these improvements were limited to a range between 0.01 and 0.14 points above baseline and were larger in ROS and SMOTE strategies.

In terms of f1-score (harmonic mean of precision and recall), the ROS implementation generated the highest improvements especially regarding underrepresented classes, with 0.09, 0.06 and 0.17 points above baseline values for Mining Pool, Mixer and Service classes, respectively (Fig. 15). Our GAN strategy followed the trend of the ROS strategy for Exchange, Gambling, Market and Mixer classes, however, for Pool and Service classes, f1-scores were only 0.02 and 0.13 points above baseline.

To investigate practical aspects regarding the implementation of different resampling techniques, Fig. 16 shows the computational cost of implementing the different techniques in terms of execution time. For state-of-the-art techniques, which do not require a training process, the time in seconds (s) they needed to create a more balanced training dataset via resampling is indicated. For GAN resampling, the time for training each GAN is now given in terms of seconds (s) for allowing a comparison. Computational costs were computed by performing resampling of the same input dataset on a server with 64GB of RAM and 16 vCPUs of 2.20 GHz.

As shown in Fig. 16, RUS and ROS techniques were the fastest strategies; they performed resampling in 8 and 17s, respectively. Meanwhile, the SMOTE algorithm needed 817s. The TL and ADASYN techniques were more costly, requiring hours for resampling the initial dataset (respectively 6,291 and 14,058s). The GAN strategies reached 1,000 epochs in a range of 409s to 521s; 25,000 epochs were reached in a range of 10,232s to 13,060s. This computational test confirmed the linear dependence of GAN training time - during the training process the execution time required to reach 25,000 epochs was about 25 times the one required to reach 1,000 epochs.

## 5.5 Discussion

In this study, an approach for using adversarial learning (GANs) to tackle class imbalance in Bitcoin data was introduced with the ultimate goal of decreasing Bitcoin entity anonymity. Our approach showed promising results that we can summarize as follows:

- All GAN architectures generated improvements in Bitcoin entity classification;
- The classifier trained with the Vanilla GAN-generated synthetic data yielded better results than classifiers trained with WGANs and unrolled GANs;
- Vanilla GAN implementations performed better using shorter training epochs;
- Vanilla GAN implementations using short training epochs achieved high repeatability;
- Vanilla GAN classifiers generated overall higher accuracy, precision, recall and f1-score than the baseline model (based on imbalanced data);

**Table 9** Classification metrics obtained by models trained with datasets resampled with different strategies. The best value of each metric is highlighted

| | Score % | prec. | recall | f1 score | MCC | AUC |
|---|---|---|---|---|---|---|
| Baseline | 94.67 | 0.91 | 0.84 | 0.87 | 0.90 | 0.98 |
| RUS | 83.10 | 0.88 | 0.83 | 0.84 | 0.74 | 0.97 |
| TL | 94.67 | 0.91 | 0.84 | 0.87 | 0.90 | 0.98 |
| ROS | 96.57 | **0.95** | **0.93** | **0.94** | **0.94** | 0.99 |
| SMOTE | 95.70 | 0.85 | 0.96 | 0.90 | 0.93 | 0.99 |
| ADASYN | 94.79 | 0.90 | 0.87 | 0.88 | 0.91 | 0.99 |
| GAN-classifier$_1$ | **96.64** | 0.94 | 0.90 | 0.92 | **0.94** | 0.99 |
| GAN-classifier$_3$ | 96.20 | 0.94 | 0.89 | 0.91 | 0.93 | 0.99 |

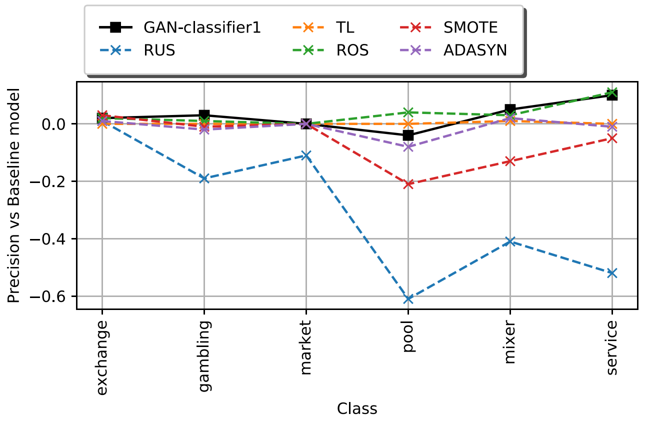


**Fig. 13** Improvements compared to baseline and breakdown in terms of precision

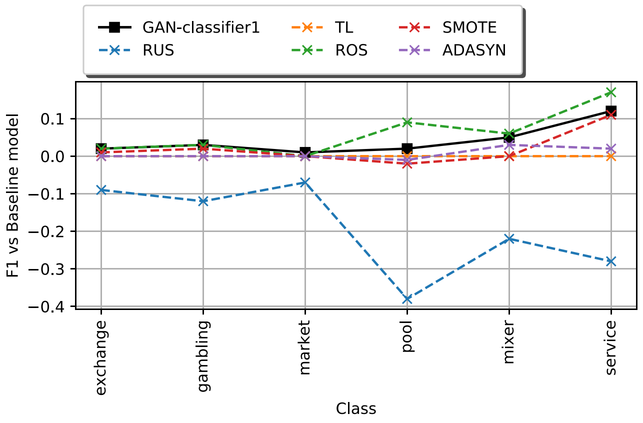


**Fig. 15** Improvements compared to baseline and breakdown in terms of f1-score

- The best GAN-based classifiers were in the top 3 techniques for classification improvements in terms of all considered metrics (overall accuracy, precision, recall, f1-score, MCC and AUC) compared to other resampling techniques (together with ROS);
- Except for the two random resampling techniques RUS and ROS, the GAN technique was the most efficient strategy in terms of single task computational costs.

## Limitations and strengths

The model trained with data generated by the Vanilla GAN architecture reached higher values of accuracy than models trained with data generated from WGANs and unrolled GANs. However, as shown in Tables 3, 4 and 5, none of the classifiers generated sufficient improvements for the Pool and Service classes (f1-scores below 0.90), regardless of the considered GAN architectures and the number of epochs. This effect motivates two hypotheses regarding possible limitations of the dataset and limitations of the GAN approach. On the one hand, the issue could be related to the distribution of those affected classes in the training dataset. In this case, after splitting into training and testing datasets, the training set may not have enough sample variety for describing all possible behaviours, which causes a loss of quality during the GAN training. On the other hand, it could happen that - although presenting a sparse distribution - affected classes are characterized by several high-density regions in the feature space. This phenomenon may affect GAN training such that only behaviours in these high-density regions are learnt.

The best values were obtained using a Vanilla GAN architecture by training this network with a lower number of epochs (1,000). In fact, too long training of this architecture (using a high number of epochs) caused a decrease of classification accuracy, probably due to known downside effects of GANs as discussed in Section 2.2 (the mode collapse effect - overfitting).

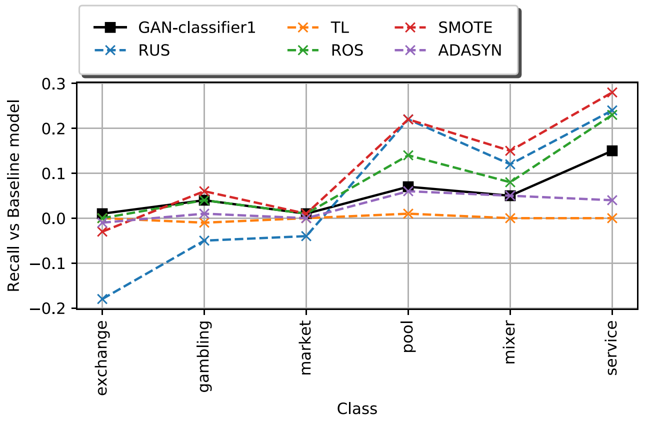


**Fig. 14** Improvements compared to baseline and breakdown in terms of recall

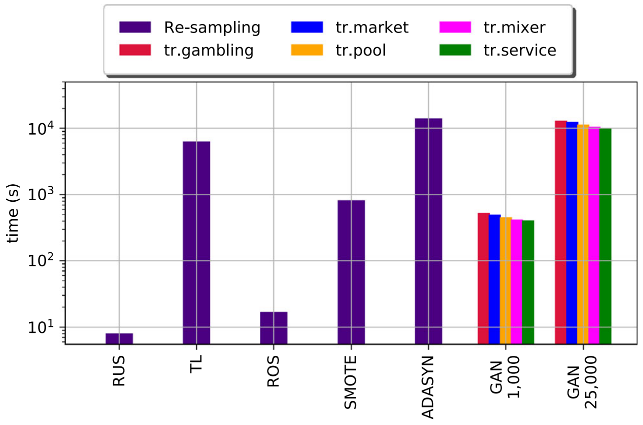


**Fig. 16** Execution times of different resampling strategies for the same input dataset. For GAN strategies, the training time is indicated

This behaviour is highlighted in Fig. 11, which shows the downward trend of obtained accuracy with increased training time related to the Vanilla GAN. The same Figure highlights how the WGAN was affected in a contrary way. Results showed that more training time (more epochs) was required for training the WGANs and for obtaining more valuable synthetic samples. However, within the range of epochs considered in the experiments, the accuracy of the model trained with the WGAN architecture was lower than the accuracy achieved by the classifier trained with the Vanilla GAN.

The two best GAN configurations we found, the Vanilla GAN trained with 1,000 and 25,000 epochs (GAN-classifier$_1$ and GAN-classifier$_3$), were tested another 5 times changing the composition of the training dataset used for training each class GAN, and they showed that the most stable (least varying) classifier with highest accuracy value was obtained by training GANs in a short period (1,000 epochs).

## Comparison to other resampling techniques

The fourth experiment demonstrated once more that the class imbalance problem is an important problem in the Bitcoin entity classification task and that it strongly affects the quality of the classifiers. All used over-sampling techniques (ROS, SMOTE and ADASYN) generated improvements in terms of recall, f1-score, MCC and AUC, while under-sampling techniques (RUS and TL) performed poorly (Table 9).

Comparing the GAN-based strategy with 5 frequently used resampling techniques, our approach represented the best solution in terms of accuracy with 96.64%; 1.97% more than the baseline accuracy. The GAN-classifier$_1$ together with ROS reached also the highest overall MCC score of 0.94. However, in terms of overall precision, recall and f1-score the ROS strategy outperformed the GAN strategy slightly.

These results, together with the results obtained in the third experiment (repeatability of the results) definitively invalidate our first hypothesis related to the limitation of the training dataset distribution. In fact, they show that simple replication of the training samples (ROS algorithms) generates good improvements in the Pool and Service classes as well (Fig. 15). However, the small improvements generated with the GAN resampled datasets confirm our second hypothesis, that GANs learn partial information of Pool and Service classes only, likely due to the sparse distribution of their behaviours and the presence of high-density regions in the feature space. This assumption is confirmed by analysing Table 9 where GAN-classifiers show high values of precision (i.e. low false positives) and low values of recall (i.e. high false negatives).

Finally, since GANs technologies help to generate samples of specific entities with high variability, they are less prone to overfitting than the ROS technique. Further, following the results obtained, this sampling variability does not strongly affect the classification performance as it is the case of the other techniques generating higher variability like SMOTE and ADASYN. This demonstrates that not only in image applications [82], but also in Bitcoin applications, GANs are able to learn the data distribution and generate new samples starting from a very limited class representation (very high imbalance ratio)

## Comparison of execution times

In terms of execution cost, the two simple resampling techniques (RUS and ROS) beat all the others in terms of computation speed (less than 20s) by a lot. However, comparing the remaining resampling techniques, the GAN strategy presents a high cumulative training time. Nevertheless, using a parallel training process, the GAN technique trained in 1,000 epochs became the next best solution after RUS and ROS, as shown by its fast-training time (average $\sim$ 460s for each implementation). The training process could be parallelized by training the 5 GANs at the same time using different threads. In fact, a single GAN task was 1.77 times faster than the whole SMOTE process, and respectively 13.7 and 30.5 faster than the TL and ADASYN algorithms.

## Comparison with previous work

In Table 10, our GAN-based classification results are compared with previous studies that directly apply Artificial Neural Network (ANN), Random Forest (RF) [83], Gradient Boosting Classifiers (GBC), GBC combined with SMOTE [7], Logistic Regression (LR) and two implementations of Light Gradient Boosting Machine (LGBM) [6, 36] in terms of overall and per class f1-score. Our approach is the best in terms of overall f1-score when using ROS (0.94) and shares second place with the Light Gradient Boosting Machine (LGBM) when using the GAN-based method. In particular, our GAN-based approach represents the best solution for detecting Exchange (0.98) and Market (0.94) entities by a difference of several points compared to previously published models. Further, it is in second place for detecting Gambling (0.95) and Service (0.89) classes (slightly outperformed by our ROS implementation for the latter), while it has problems detecting Pool entities. More specifically, the four classes in which our GAN approach

**Table 10** Comparison of per-class and overall f1-score achieved by comparable state-of-the-art works. The best value, for each class, is highlighted

| | f1-score | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | [83] | [7] | [6] | [36] | Proposed method | | | | |
| Class | RF | ANN | GBC | GBC (SMOTE) | LR | LGBM | LGBM | ROS | GAN classifier$_1$ |
| Exchange | 0.78 | 0.56 | 0.86 | 0.84 | 0.91 | 0.92 | 0.89 | **0.98** | **0.98** |
| Gambling | 0.77 | 0.48 | 0.78 | 0.78 | 0.82 | **0.97** | 0.83 | 0.94 | 0.95 |
| Market | - | - | 0 | 0 | - | - | 0.78 | 0.98 | **0.99** |
| Pool | 0.86 | 0.65 | 0.90 | 0.86 | 0.67 | 0.67 | 0.83 | **0.91** | 0.82 |
| Mixer | 0.82 | 0.45 | 0.33 | 0.97 | - | - | **0.98** | 0.89 | 0.88 |
| Service/Other* | - | - | 0.22* | 0.08* | 0.87 | 0.88 | - | **0.92** | 0.89 |
| Overall | - | - | 0.75 | 0.76 | 0.87 | 0.92 | 0.87 | **0.94** | 0.92 |

performs better are also the ones that are interesting from an investigation point of view since those four, together with Mixers, represent the entities more prone to be involved in illicit transactions such as money laundering (Section 2.1). It is to be noted that our ROS and GAN-based approaches together outperform all other models for the majority of classes except LGBM for the Gambling and Mixer classes. Further, our implementations generate overall precision, recall and f1-score scores that are aligned with the results presented in [70] (between 0.90 and 0.94), however, we achieve better overall accuracy ($\sim$ 5%), and better recall values per class, especially for Exchange, Mining, Mixer and Gambling classes.

Finally, comparing our GAN-based approach with work presented in [78], our metrics exceed their results (with respect to WCGAN, DRAGAN and SMOTE) in terms of AUC ($\sim$ 0.07) and recall ($\sim$ 0.59), even though precision decreases slightly ($\sim$ 0.05).

## 6 Conclusions

Bitcoin entity classification is an important task in cryptocurrency network analysis for decreasing entity anonymity, for detecting classes related to abnormal or illicit activities and for increasing the network's resilience to cyber-attacks (detecting the more vulnerable classes). Typically, this task is performed by applying supervised machine learning algorithms whose results are strongly conditioned by ground truth Bitcoin datasets. Yet, these labelled input datasets usually do not contain a homogeneous population of the various Bitcoin entity classes, yielding a class imbalance problem, which results in poor classification performance.

Current blockchain studies are transferring a variety of technologies that have already shown good results in other domains to the Bitcoin blockchain domain. Following this trend, in this work, we introduced the application of Generative Adversarial Networks (GANs) to address the Bitcoin imbalance problem with the aim to evaluate how this synthetic information can improve the entity classification task. GANs have mainly been used in the image or video processing domain and specifically for addressing imbalance problems. In this work, we investigated different types of GAN architectures and how we can apply them to the Bitcoin domain in order to address the imbalance problem and improve the final multi-class classification. Experiments were performed to check the repeatability of GAN-based resampling and to compare our approach to other common resampling techniques.

Our GAN-based classification approach generally obtained promising results, achieving the best results in terms of accuracy and was among the best in terms of precision, recall and f1-score compared to other resampling techniques. Furthermore, the presented methods outperformed previously published models with regard to Exchange, Market, Pool and Service entity classification.

Along the way, we detailed and highlighted here GAN-specific characteristics such as the influence of training epochs, which need to be taken into consideration when applying GANs to Bitcoin-related data. Finally, we detailed the limitations of this approach when the training dataset is composed of dense areas in the features space. In these cases, such distributions seem to force GANs to learn certain behaviours and forget others.

This study represents a first step towards the usage of GAN technologies with Bitcoin behavioural data. Motivated by our promising results, future work could be directed towards testing other optimization functions (like Adagrad [84] or Adadelta [85]), evaluating changes in G and D networks. In this way, it will be possible to evaluate if with the help of a more robust optimizer the GANs will learn to generate more valuable samples (in terms of classification improvements). Further, it will be interesting to implement multi-GAN solutions for each class, i.e. training more than one GAN for each class in order to increase the knowledge about the training dataset and evaluate their effects in generating information. Based on the promising results presented in this paper, future studies could focus on analysing alternative data augmentation techniques such as Mixup [86] for Bitcoin entity classification. Finally, it could be interesting to apply and validate the introduced approach to address class imbalance problems using other machine learning models different from Random Forest, for example by including Neural Networks or tensor-based classifiers [87]. Overall, our study has shown that by applying carefully selected resampling techniques the Bitcoin class imbalance problem can indeed be tackled, leading to very good classification results across a broad range of critical (and often under-represented) Bitcoin entity classes, which may ultimately impact positively on Bitcoin de-anonymization and the detection of illicit activities within the network.

**Acknowledgements** This work has been partially supported by the Spanish Centre for the Development of Industrial Technology (CDTI) under the project ÉGIDA (CER20191012) - RED DE EXCELENCIA EN TECNOLOGIAS DE SEGURIDAD Y PRIVACIDAD.

**Author Contributions** F. Zola: Conceptualization of this study, Related Work, Methodology, Results, Discussion, Writing - Original draft preparation
L. Segurola-Gil: Related Work, Data curation,
J.L. Bruse: Discussion, Writing - Review and editing
M. Galar: Discussion, Writing - Review and editing
R. Orduna-Urrutia: Conceptualization of this study, Supervision

**Funding** Not applicable

## Declarations

**Availability of Data and Material (Data Transparency)** Not applicable

**Code Availability** Not applicable

**Competing interests** (check journal-specific guidelines for which heading to use)

**Conflict of Interests** Not applicable

## References

1. Nakamoto S (2019) Bitcoin: A peer-to-peer electronic cash system. Technical Report, Manubot
2. Foley S, Karlsen JR, Putniņš TJ (2019) Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? Rev Financ Stud 32(5):1798–1853
3. Marella V, Upreti B, Merikivi J, Tuunainen VK (2020) Understanding the creation of trust in cryptocurrencies: the case of bitcoin. Electron Mark:1–13
4. Saez M (2020) Blockchain-enabled platforms: Challenges and recommendations. Int J Interact Multimed Artif Intell 6(3)
5. Zola F, Bruse JL, Eguimendia M, Galar M, Orduna Urrutia R (2019) Bitcoin and cybersecurity: temporal dissection of blockchain data to unveil changes in entity behavioral patterns. Appl Sci 9(23):5003
6. Jourdan M, Blandin S, Wynter L, Deshpande P (2018) Characterizing entities in the bitcoin blockchain. In: IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, pp 55–62
7. Harlev MA, Sun Yin H, Langenheldt KC, Mukkamala R, Vatrapu R (2018) Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In: Proceedings of the 51st Hawaii international conference on system sciences
8. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Learning from imbalanced data sets. Springer
9. Monamo PM, Marivate V, Twala B (2016) A multifaceted approach to bitcoin fraud detection: Global and local outliers. In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp 188–194
10. Zheng W, Zhao H (2020) Cost-sensitive hierarchical classification for imbalance classes. Appl Intell 50(8):2328–2338
11. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Algorithm-level approaches. In: Learning from imbalanced data sets. Springer, pp 123–146
12. Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ (2016) Training deep neural networks on imbalanced data sets. In: 2016 international joint conference on neural networks (IJCNN). IEEE, pp 4368–4374
13. Manju N, Harish BS, Nagadarshan N (2020) Multilayer feedforward neural network for internet traffic classification. Int J Interact Multim Artif Intell 6(1):117–122
14. Alotaibi A (2020) Deep generative adversarial networks for image-to-image translation: a review. Symmetry 12(10):1705
15. Brock A, Donahue J, Simonyan K (2018) Large scale gan training for high fidelity natural image synthesis. In: International conference on learning representations
16. Vondrick C, Pirsiavash H, Torralba A (2016) Generating videos with scene dynamics. Adv Neural Inf Proces Syst 29
17. Bowles C, Chen L, Guerrero R, Bentley P, Gunn R, Hammers A, Dickie DA, Hernández MV, Wardlaw J, Rueckert D (2018) Gan augmentation: Augmenting training data using generative adversarial networks. arXiv:1810.10863
18. Abusitta A, Aïmeur E, Abdel Wahab O (2020) Generative adversarial networks for mitigating biases in machine learning systems. In: ECAI 2020. IOS Press, pp 937–944
19. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
20. Wang K, Gou C, Duan Y, Lin Y, Zheng X, Wang F-Y (2017) Generative adversarial networks: introduction and outlook. IEEE/CAA J Autom Sin 4(4):588–598

21. Shamsolmoali P, Zareapoor M, Wang R, Jain DK, Yang J (2019) G-ganisr: Gradual generative adversarial network for image super resolution. Neurocomputing 366:140–153
22. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z et al (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4681–4690
23. Alqahtani H, Kavakli-Thorne M, Kumar G (2019) Applications of generative adversarial networks (gans): an updated review. Arch Comput Methods Eng:1–28
24. Kethineni S, Cao Y (2019) The rise in popularity of cryptocurrency and associated criminal activity. Int Crim Justice Rev:1057567719827051
25. Hu M, Chen J, Gan W, Chen C-M (2021) A jumping mining attack and solution. Appl Intell 51(3):1367–1378
26. Fanusie Y, Robinson T (2018) Bitcoin laundering: an analysis of illicit flows into digital currency services. Center on Sanctions & Illicit Finance memorandum
27. Sun X, Yang T, Hu B (2021) Lstm-tc: Bitcoin coin mixing detection method with a high recall. Appl Intell:1–14
28. Conti M, Kumar ES, Lal C, Ruj S (2018) A survey on security and privacy issues of bitcoin. IEEE Commun Surv Tutorials 20(4):3416–3452
29. Zola F, Eguimendia M, Bruse JL, Urrutia RO (2019) Cascading machine learning to attack bitcoin anonymity. In: IEEE International conference on blockchain (Blockchain). IEEE, pp 10–17
30. Yin HS, Vatrapu R (2017) A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning. In: IEEE International conference on big data (Big Data). IEEE, pp 3690–3699
31. Hu M, Chen J, Gan W, Chen C-M (2020) A jumping mining attack and solution. Appl Intell 51(3):1367–1378
32. Kim S, Kim B, Kim HJ (2018) Intrusion detection and mitigation system using blockchain analysis for bitcoin exchange. In: Proceedings of the international conference on cloud computing and internet of things, pp 40–44
33. Zhang Y, Wang J, Luo J (2020) Heuristic-based address clustering in bitcoin. IEEE Access 8:210582–210591
34. Paquet-Clouston M, Haslhofer B, Dupont B (2019) Ransomware payments in the bitcoin ecosystem. J Cybersecur 5(1):tyz003
35. Haslhofer B, Karl R, Filtz E (2016) O bitcoin where art thou? insight into large-scale transaction graphs. In: SEMANTiCS (Posters, Demos, SuCCESS)
36. Lin Y-J, Wu P-W, Hsu C-H, Tu I-P, Liao S (2019) An evaluation of bitcoin address classification based on transaction history summarization. In: IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, pp 302–310
37. Liao K, Zhao Z, Doupé A, Ahn G-J (2016) Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In: APWG Symposium on Electronic Crime Research (eCrime). IEEE, pp 1–13
38. Farnia F, Ozdaglar A (2020) Do gans always have nash equilibria? In: International conference on machine learning. PMLR, pp 3029–3039
39. Yuan W, Hu F, Lu L (2021) A new non-adaptive optimization method: stochastic gradient descent with momentum and difference. Appl Intell:1–15
40. Sun R, Fang T, Schwing A (2020) Towards a better global loss landscape of gans. Adv Neural Inf Process Syst 33
41. Goodfellow I (2016) Nips 2016 tutorial: Generative adversarial networks. arXiv:1701.00160
42. Dai Y, Wang S, Chen X, Xu C, Guo W (2020) Generative adversarial networks based on wasserstein distance for knowledge graph embeddings. Knowl-Based Syst 190:105165
43. Martin A, Lon B (2017) Towards principled methods for training generative adversarial networks. In: NIPS Workshop on adversarial training. In review for ICLR, vol 2016
44. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, vol 70. PMLR, pp 214–223
45. Nagarajan V, Kolter JZ (2017) Gradient descent gan optimization is locally stable. In: Advances in neural information processing systems, pp 5585–5595
46. Metz L, Poole B, Pfau D, Sohl-Dickstein J (2017) Unrolled generative adversarial networks. In: 5th International conference on learning representations, conference track proceedings. OpenReview.net
47. Sahu S, Gupta R, Espy-Wilson C (2018) On enhancing speech emotion recognition using generative adversarial networks. In: INTERSPEECH
48. Yi X, Walia E, Babyn P (2019) Generative adversarial network in medical imaging: a review. Med Image Anal 58:101552
49. Ali-Gombe A, Elyan E (2019) Mfc-gan: class-imbalanced dataset classification using multiple fake class generative adversarial network. Neurocomputing 361:212–221
50. Dai X, Yuan X, Wei X (2021) Data augmentation for thermal infrared object detection with cascade pyramid generative adversarial network. Appl Intell:1–15
51. Liu Q-M, Jia R-S, Liu Y-B, Sun H-B, Yu J-Z, Sun H-M (2021) Infrared image super-resolution reconstruction by using generative adversarial network with an attention mechanism. Appl Intell 51(4):2018–2030
52. Zhang F, Ma Y, Yuan G, Zhang H, Ren J (2021) Multiview image generation for vehicle reidentification. Appl Intell:1–18
53. Zong X, Chen Z, Wang D (2021) Local-cyclegan: a general end-to-end network for visual enhancement in complex deep-water environment. Appl Intell 51(4):1947–1958
54. Chen Y, Zhang H, Liu L, Chen X, Zhang Q, Yang K, Xia R, Xie J (2021) Research on image inpainting algorithm of improved gan based on two-discriminations networks. Appl Intell 51(6):3460–3474
55. Chen S, Chen S, Guo Z, Zuo Y (2019) Low-resolution palmprint image denoising by generative adversarial networks. Neurocomputing 358:275–284
56. Li Y, Zhang Y, Yu K, Hu X (2021) Adversarial training with wasserstein distance for learning cross-lingual word embeddings. Appl Intell:1–13
57. Yang Z, Chen W, Wang F, Xu B (2018) Generative adversarial training for neural machine translation. Neurocomputing 321:146–155
58. Athanasiadis C, Hortal E, Asteriadis S (2019) Audio–visual domain adaptation using conditional semi-supervised generative adversarial networks. Neurocomputing
59. Merino T, Stillwell M, Steele M, Coplan M, Patton J, Stoyanov A, Deng L (2019) Expansion of cyber attack data from unbalanced datasets using generative adversarial networks. In: International conference on software engineering research, management and applications. Springer, pp 131–145
60. Yilmaz I, Masum R (2019) Expansion of cyber attack data from unbalanced datasets using generative techniques. arXiv:1912.04549
61. Mukhtar N, Batina L, Picek S, Kong Y (2021) Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices
62. Wang P, Li S, Ye F, Wang Z, Zhang M (2020) Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, pp 1–7

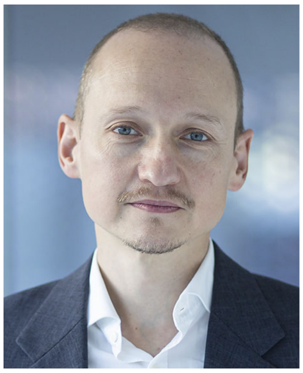
63. Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G (2017) Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 73:220–239
64. García V, Sánchez JS, Marqués AI, Florencia R, Rivera G (2019) Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. Expert Syst Appl:113026
65. Pereira RM, Costa YandreMG, Silla Jr C N (2020) Mltl: A multi-label approach for the tomek link undersampling algorithm. Neurocomputing 383:95–105
66. Fernandez A, Garcia S, Herrera F, Chawla NV (2018) Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res 61:863–905
67. Vo MT, Nguyen T, Vo HA, Le T (2021) Noise-adaptive synthetic oversampling technique. Appl Intell:1–10
68. Oussidi A, Elhassouny A (2018) Deep generative models: Survey. In: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). IEEE, pp 1–8
69. Xie Y, Zhang T (2018) Imbalanced learning for fault diagnosis problem of rotating machinery based on generative adversarial networks. In: 2018 37th Chinese Control Conference (CCC). IEEE, pp 6017–6022
70. Nerurkar P, Bhirud S, Patel D, Ludinard R, Busnel Y, Kumari S (2021) Supervised learning model for identifying illegal activities in bitcoin. Appl Intell 51(6):3824–3843
71. Bartoletti M, Pes B, Serusi S (2018) Data mining for detecting bitcoin ponzi schemes. In: Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, pp 75–84
72. Ranshous S, Joslyn CA, Kreyling S, Nowak K, Samatova NF, West CL, Winters S (2017) Exchange pattern mining in the bitcoin transaction directed hypergraph. In: International conference on financial cryptography and data security. Springer, pp 248–263
73. Liang J, Li L, Luan S, Gan L, Zeng D (2019) Bitcoin exchange addresses identification and its application in online drug trading regulation. In: 23rd Pacific Asia Conference on Information Systems: Secure ICT Platform for the 4th Industrial Revolution, PACIS 2019
74. Monamo MP (2018) Anomaly detection in the open financial markets: A case for the bitcoin network. University of Johannesburg, South Africa
75. Pfenninger M, Rikli S, Bigler DN (2021) Wasserstein gan: Deep generation applied on financial time series. Available at SSRN 3877960
76. Grilli L, Santoro D (April 2020) Generative Adversarial Network for Market Hourly Discrimination. In: 3RD International conference on mathematical and related sciences: current trends and developments proceedings book
77. Zola F, Bruse JL, Barrio XE, Galar M, Urrutia RO (2020) Generative adversarial networks for bitcoin data augmentation. In: 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). IEEE, pp 136–143
78. Han J, Woo J, Hong JW-K (2020) Oversampling techniques for detecting bitcoin illegal transactions. In: 21st Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, pp 330–333
79. Toyoda K, Ohtsuki T, Mathiopoulos PT (2018) Multi-class bitcoin-enabled service identification based on transaction history summarization. In: IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, pp 1153–1160
80. Zou F, Shen L, Jie Z, Zhang W, Liu W (2019) A sufficient condition for convergences of adam and rmsprop. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 11127–11135
81. Chicco D, Jurman G (2020) The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC Genomics 21(1):1–13
82. Sampath V, Maurtua I, Martín JJA, Gutierrez A (2021) A survey on generative adversarial networks for imbalance problems in computer vision tasks. J Big Data 8(1):1–59
83. Lee C, Maharjan S, Ko K, Woo J, Hong JW-K (2020) Machine learning based bitcoin address classification. In: International conference on blockchain and trustworthy systems. Springer, pp 517–531
84. Lydia A, Francis S (2019) Adagrad–an optimizer for stochastic gradient descent. Int J Inf Comput Sci 6(5)
85. Dogo EM, Afolabi OJ, Nwulu NI, Twala B, Aigbavboa CO (2018) A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In: 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS). IEEE, pp 92–99
86. Liang D, Yang F, Zhang T, Yang P (2018) Understanding mixup training methods. IEEE Access 6:58774–58783
87. Hu C, He S, Wang Y (2021) A classification method to detect faults in a rotating machinery based on kernelled support tensor machine and multilinear principal component analysis. Appl Intell 51(4):2609–2621

**Francesco Zola** obtained his Bachelors degree in Telecommunication Engineering at University of Cassino and Southern Lazio, Italy, in 2012 and his Master's degree in Computer Science in the same university, in 2015. He works as associate researcher and data scientist in Vicomtech in the department of Digital Security applying machine learning in cybersecurity projects. In 2019 he started his doctorate in collaboration with the Public University of Navarre (UPNA), focusing his works on graph analysis, machine learning, data augmentation and classification. Francesco is involved in several European and industrial projects related to cybersecurity, blockchain analysis and anomaly detection.

**Lander Segurola-Gil** is a mathematician who obtained his Bachelors degree in the University of País Vasco Euskal Herriko Unibertsitatea (UPV-EHU) and received a Masters degree in Mathematics and Applications from the University of Madrid (UAM). Since 2019, he works as a research assistant in Vicomtech in the department of Digital Security. His research interests are Naive Bayes networks, attack detection and machine learning applied to time series.

**Dr. Jan L. Bruse** graduated in Mechanical Engineering at RWTH Aachen University, Germany, in 2013 and received his doctorate in Biomedical Engineering from University College London, United Kingdom, in 2017. His PhD thesis combines Medical Image Analysis, Computational Simulation and Machine Learning for the development of Clinical Decision Support Systems. Jan has participated in several international and interdisciplinary research collaborations, has multiple peer-reviewed publications in journals and congresses indexed in the engineering and clinical sector and is reviewer for several international journals. Since September 2017, Jan is Research Engineer at Vicomtech, Spain, in the area of Data Intelligence for Energy and Industrial Processes where he develops Artificial Intelligence (AI) and Machine Learning platforms within the context of Industry 4.0 projects.

**Dr. Mikel Galar** received the MSc and PhD degrees in Computer Science in 2009 and 2012, both from the Public University of Navarre (UPNA), Spain. He is currently an associate professor at the Department of Statistics, Computer Science and Mathematics at the UPNA. He is the author of 35 published original articles in international journals and more than 50 contributions to conferences. He is also reviewer of more than 35 international journals. His research interests are machine learning, data mining, classification, fuzzy systems and big data. He is a member of the IEEE, the European Society for Fuzzy Logic and Technology (EUSFLAT) and the Spanish Association of Artificial Intelligence (AEPIA). He has received the extraordinary prize for his PhD thesis from the Public University of Navarre and the 2013 IEEE Transactions on Fuzzy System Outstanding Paper Award for the paper "A New Approach to Interval-Valued Choquet Integrals and the Problem of Ordering in Interval-Valued Fuzzy Set Applications" (bestowed in 2016).



**Dr. Raul Orduna-Urrutia** received his degree in Computer Engineering at the Faculty of Informatics of San Sebastian by the University of the Basque Country (UPV/EHU) and obtained a PhD degree in Computer Science and Artificial Intelligence at the School of Industrial and Telecommunications Engineering (ETSIIT) of the Public University of Navarre (UPNA). He is currently an associate professor at the Department of Statistics and Computer Science at the UPNA, and, at the same time, the Digital Security Director in Vicomtech. He has taken part or led projects related with ethical hacking, forensic analysis, malware analysis, access control and cryptography. He has participated in more than 6 successful funded projects, is the author of 6 published original articles in international journals. He is also a reviewer of Fuzzy Sets and Systems.

## Affiliations

**Francesco Zola**[1,2] · **Lander Segurola-Gil**[1] · **Jan L. Bruse**[1] · **Mikel Galar**[2] · **Raul Orduna-Urrutia**[1]

Lander Segurola-Gil
lsegurola@vicomtech.org

Jan L. Bruse
jbruse@vicomtech.org

Mikel Galar
mikel.galar@unavarra.es

Raul Orduna-Urrutia
rorduna@vicomtech.org

1 Digital Security, Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Paseo Mikeletegi, Donostia/San Sebastian, 20009, Spain

2 Institute of Smart Cities, Public University of Navarre, Pamplona, 31006, Spain