A Tale of Two Methods: Unveiling the limitations of GAN and the Rise of Bayesian Networks for Network Traffic Data Generation

Anonymous Author(s)

ABSTRACT

Network Intrusion Detection Systems (NIDS) are now widely used for network traffic monitoring. However, an enduring challenge persists in evaluating their performance, because such evaluation requires legitimate and attack traffic. In this paper, we focus on synthetically generating legitimate traffic as acquiring high-quality legitimate traffic remains a daunting task in practice. Recent contributions involve machine learning-driven approaches, notably through Generative Adversarial Networks (GAN). However, evaluations of GAN-generated data often disregards pivotal attributes, such as protocol adherence. Our study addresses the gap by proposing an encompassing set of quality metrics that assess the quality of synthetic legitimate network traffic. In addition, we empirically compare advanced network-oriented GANs with a more straightforward yet powerful generative model, Bayesian Networks (BN). According to our proposed evaluation metrics, BN-based network traffic generation outperforms state-of-the-art GAN-based methods, yielding substantially more realistic and viable synthetic benign traffic, all while minimizing computational costs.

1 INTRODUCTION

Network Intrusion Detection Systems (NIDS) play a vital role in ensuring network security. Evaluating the performance of NIDS is imperative and involves the need for a significant amount of legitimate (benign) data reflecting user activity as well as data containing traces of attacks. In this paper, our focus is specifically on benign traffic, which is instrumental in assessing the occurrence of false positives generated by a NIDS.

Using authentic benign data is generally infeasible due to privacy concerns, preventing its sharing. Even if sharing were possible, there remains a challenge regarding data labeling, as attacks may occur during capture and may not be easily discernible as such [22]. To overcome these challenges, various approaches [27, 35, 40, 41] have attempted to tackle the issue by simulating benign traffic. These endeavors involve replicating a user's actions within a controlled environment. However, this simulation-based approach is computationally intense thus unscalable to large-scale use. This type of methods thus remain inflexible across diverse user scenarios [1].

In recent years, there has been a growing interest in an alternative approach, specifically the use of machine learning (ML) techniques to generate synthetic benign traffic. These data generation methods —widely successful in creating diverse types of data such as images [4, 13], textual data [36], and tabular data [9, 15]— have inspired researchers to adapt these techniques for generating synthetic network traffic data. Notably, Generative Adversarial Networks (GAN) [13] have been applied in numerous studies to create network data [8, 14, 17, 20, 23, 25, 32, 39, 46]. Other approaches have explored the use of Auto-Regression models [50] and sequential data generators [19, 51].

However, there is inconsistency in the evaluation metrics applied to assess the quality of generated synthetic data [6, 30, 36]. Different data generation methods present their results using diverse data quality evaluation metrics, making it challenging to systematically compare the quality of synthetic data produced by various methods.

While these studies have introduced various criteria to evaluate effective data generation, defining a systematic set of metrics that comprehensively covers different aspects of synthetic data quality remains highly demanding and is identified as a gap in previous research works. For instance, the concept of "authenticity" remains unexplored, as outlined by Alaa et al. [2], despite its potential to mitigate severe privacy breaches. Neglecting essential criteria, as cautioned by [6], can undermine the overall quality of data generation.

We address this challenge by making the following contributions:

- We define a comprehensive set of criteria for assessing the quality of generated traffic. Our proposed evaluation framework includes 8 metrics specifically crafted to evaluate these criteria, with a focus on network traffic generation.
- Based on the proposed metrics, we advocate the necessity of capturing explicitly the probabilistic dependency between different network attributes to produce high quality synthetic network traffic data. We hence propose Bayesian Networks (BNs) as a computationally efficient yet effective solution for network traffic generation. We conduct a comparative analysis across three widely used network traffic datasets, pitting the BN-based approach against state-of-the-art methods, including those based on GANs. This comparison is carried out using our tailored evaluation framework.

Due to space constraints, this paper does not aim to present results related to the use of the generated data for assessing the false positives generated by a NIDS. The primary focus of this paper is on the contributions outlined above.

The paper is structured as follows. Section 2.3 will introduce the criteria governing effective data generation. In Section 3, we will unveil a comprehensive set of scoring functions which, to the best of our knowledge, constitutes the most exhaustive and comprehensible toolkit for evaluating the quality of network traffic generation. Drawing insights from [5, 12], an assessment of all the criteria for successful generation highlights the inadaptability of GANs for generating tabular data. Consequently, Section 4 will introduce a novel generation method based on Bayesian Networks. Subsequently, in Section 5, we will demonstrate the superiority of these models. Bayesian Networks emerge as not only more straightforward and interpretable models but also exhibit faster training times and superior overall data generation quality compared to GANs.

2 RELATED WORKS

When addressing network traffic generation, some studies opt to create synthetic network packets [8, 11, 46], while others chose

to generate network traffic flows [14, 23, 34, 51] or network features [17, 19, 39]. As a large proportion of the Internet traffic is nowadays encrypted, we focus here on Network Flow generation, leaving aside packet (i.e., payload) generation.

2.1 Network flow data structure

A *network flow* is a sequence of packets from one computer source to a destination. Network flows are identified by five features: *source IP address*, *destination IP address*, *source port*, *destination port*, and *transport protocol*. Tools such as Zeek [45] and nProbe [44] can monitor a network and extract from the packets some other features such as the number of bytes, number of packets, or duration of a network flow. However, depending on the tool, the set of features describing a network flow may not be the same. Sustained endeavors have been made to establish a standardized feature set, yet a unanimous consensus remains elusive [37]. In this paper, network flow generation means the generation of the five features identifying a network flow and of some other features describing the flow. We use three different feature sets from a minimal one with 8 features to a large one with 30 features (see Subsection 5.1 for more details).

Network flows can be stored in a unidirectional or bidirectional format. Depending on the extracted features, it is possible to convert network flows from one format to the other. In our work, we focus on unidirectional flow generation and use datasets that either directly store unidirectional flows or store bidirectional flows but with features that permit the conversion to unidirectional flows. As elucidated by Bourou et al. [7], network flow generation can be viewed as tabular data generation: each unidirectional flow forms a row, and the features constitute the columns.

2.2 ML-based generation of network traffic

Since 2017, the application of GAN to network flow generation has gathered significant attention, notably pioneered by Ring et al. [34]. The landscape of research in this area can be categorized into two groups. The first group directs its efforts towards harnessing GAN to *amplify specific malicious traffic* classes within datasets. The objective of this data augmentation is twofold: to enhance NIDS capacities and to engender novel malicious samples capable of eluding existing NIDS [16, 20, 32]. In contrast, the goal of the second group of work is to *generate benign traffic* that closely mimics captured network traffic [7, 23, 34, 51].

Yet, as discussed in [5, 42], the appropriateness of deep neural networks, including GAN, for categorical tabular data is somewhat limited due to their intrinsic design and optimization for continuous numerical data distributions. Alternatively, classical statistical methodologies, despite their simplicity, often yield superior generation outcomes [12]. Within this study, we aim to investigate the viability of employing Bayesian Networks (BN) [15] for generating synthetic benign network flows.

Beyond the realm of GAN, it is noteworthy to highlight two other studies that follow distinct approaches to network flow generation. Firstly, STAN [50] introduces an autoregressive model that captures the distribution of network flow traffic using the UGR'16 dataset [22]. Secondly, Song et al. [43] introduce a system capable of learning and replicating network host behavior to synthesize statistical network flow features. These works underscore the significance of capturing statistical correlations between network traffic attributes in generating informative network traffic data. This reinforces our conviction in the potential of BN models as a means to generate network flows.

2.3 Quality evaluation of data generation

The assessment of synthetic data has been explored across various domains beyond cybersecurity [6, 36, 52]. Acknowledging that the evaluation task cannot be encapsulated by a single metric [6], it is evident that effective generation involves not only producing high-quality samples but also generating a diverse array of samples. In this section, we delve into the evaluation methodologies employed in the state of the art, starting with an examination of *tabular data* generation, followed by an exploration of *network traffic* generation.

2.3.1 Evaluation of tabular data generation. As explained in Section 2.1, the generation of network flows can be viewed as the generation of tabular data. Consider a scenario wherein the objective is to synthesize a set of observations for two features, denoted as X_{gen} and Y_{gen}, which are intended to adhere to the joint distribution (X_{source}, Y_{source}). To evaluate such tabular generation, a logical approach involves first comparing the *marginal distributions*—specifically, contrasting $P(X_{gen})$ with $P(X_{source})$ and $P(Y_{gen})$ with $P(Y_{source})$.

Marginal distributions are often compared feature-wise by using established metrics like Jensen-Shannon Divergence (JSD), Earth Mover's Distance (EMD), or the Kolmogorov–Smirnov test (KSTest) [5, 12, 30]. However, it is essential to note that juxtaposing marginal distributions independently is insufficient: preserving the *inter-variable correlations* present in real datasets is equally crucial [17, 19, 24, 29]. This can be achieved by comparing Pairwise Correlation Matrices [12], or employing a Chi-squared Test [30].

Evaluation of generated tabular data can also encompass the examination of sample distributions, akin to comparing *joint distributions*—specifically, evaluating $P(X_{gen},Y_{gen})$ against $P(X_{source}, Y_{source})$ [2, 28, 53]. This evaluation of sample distribution predominantly follows two approaches: one inspired by *precision and recall* metrics [6, 21], and the other revolving around the assessment of the *utility* of generated samples in subsequent machine learning tasks [54]. Furthermore, the generation of data should *transcend mere replication* of real data, yielding new samples that genuinely adhere to the real distribution [2, 6, 12].

In the general case, a good generation approach should follow the below three criteria [38]:

- **Realism**: a synthetic sample should be sampled from the same distribution as the real data.
- **Diversity**: the distribution of the generated samples should have the same variability as the real data.
- **Novelty** (called authenticity in [38]): a generated sample should be sufficiently different from the samples of the real distribution. To avoid any confusion with the classical definition of authenticity in security, we prefer the term "**novelty**".

In Table 1, we have collected the metrics used for evaluating tabular data generation, and provided a brief description of their application scale as well as the potential above-mentioned criteria they assess.

Score	Criteria			Description of the score		
	Real.	Div.	Nov.	Input	Description	Ref.(s)
Jensen-Shannon Divergence (JSD)	✓	✓		Marg. Distr.	Jensen Shannon Divergence between generated and real features.	[4, 6]
Wasserstein-1 (EMD)	✓	✓		Marg. Distr.	Wasserstein-1 Distance between generated and real features.	[6]
Kolmogorv-Smirnov test (KS test)	✓	✓		Marg. Distr.	Distance between the generated feature distribution and the real feature's cumulated distribution function (CDF).	[5, 30]
Chi-squared test	✓			Marg. Distr.	Verification that dependencies present across real features are present across synthetic features.	[30]
Pairwise Correlation Difference	✓			Joint Distr.	Compare the pairwise correlation matrix of the generated data with the real one.	[12, 24, 52]
Precision	✓			Joint Distr.	Probability that a generated sample belongs to the real distribution.	[6, 21]
Recall		✓		Joint Distr.	Probability that a real sample belongs to the generated distribution.	[6, 21]
Coverage		✓		Joint Distr.	Probability mass of the real distribution covered by the generated distribution.	[6, 28]
Density	✓			Joint Distr.	Probability mass of the generated distribution covered by the real distribution.	[6, 28]
LogCluster	✓	✓		Joint Distr.	Proximity of real and generated samples, evaluated through clusterization.	[12, 48]
MSID	✓	✓		Joint Distr.	Geometrical similarity between real and generated data manifolds.	[6]
TSTR	✓	✓		Joint Distr.	Difference of performance between a classifier trained on real data and another one on synthetic data.	[6, 12, 30, 54]
Global Data-Copying Test			✓	Sample	Statistical test that verifies if a generated sample is a copy from a training sample.	[2, 6, 26]
Membership disclosure			✓	Joint Distr.	Resistance toward a specific membership inference attack.	[12, 29]
Tri-dimensional evaluation set	✓	✓	✓	Sample	Adaptation of precision and recall to the support of distributions.	[2]

Table 1: The main evaluation methods for the quality assessment of synthetic tabular data, and related criteria. Real.: Realism, Div.: Diversity, Nov.: Novelty

2.3.2 Evaluation of network traffic data generation. The evaluation methodology for network flow generation mirrors the one proposed for tabular data. This process involves employing metrics such as JSD [17, 50], EMD [7, 19, 23, 39, 47, 51], or gauging the performance of a downstream classification model [7, 19, 46, 50, 51, 53].

Some articles [34, 39, 50] put forth what Ring et al. call a *Domain Knowledge Check*, wherein the generated network flows are assessed for conformity to fundamental network protocol regulations. This check is related to the *Realism* criterion but they are not equivalent. For example, a network flow comprising only one packet has a duration of 0 (as it is the sum of inter-arrival times between packets); if a model generates a network flow that comprises only one packet and has a duration of 0.001 seconds, the value of duration is close to the expected value but it still deviates from the network protocol rules. This example highlights the necessity to evaluate network traffic using a distinct criterion—one that encapsulates the notion of adherence to specific network rules. We thus introduce a distinct criterion termed *Compliance* to echo this need, which quantifies the degree to which a generated flow aligns with network protocol specifications. It is important to highlight that while *Compliance* and *Realism* share some commonalities, they do not encompass precisely the same properties. *Realism* gauges the extent to which a generated sample conforms to the distribution of real data, whereas *Compliance* scrutinizes the property of a sample concerning the specifications mandated by network protocols.

We have consolidated all the metrics and the corresponding criteria they evaluate in Table 2. Additionally, we specify the nature of their input and the output they yield.

2.3.3 Bottlenecks in the current evaluation methods. Upon comparing Table 1 and Table 2, it becomes evident that certain aspects evaluated in the context of tabular data generation, as depicted in Table 1, do not find representation in Table 2. Consequently, we are prompted to identify deficiencies in the existing evaluation practices for network flow generation.

As discernible from Table 2, the assessment of whether correlations between attributes in real data are faithfully retained in synthetic data frequently goes unheeded. Merely a handful of studies [7, 19, 27, 47] underscore the significance of upholding correlations within generated network traffic flows through a dedicated evaluation. This aspect holds such importance that it was advocated that evaluations of network flow generators prioritize this dimension [27].

Furthermore, a current evaluation shortfall involves novelty assessment. In network flow generation, only DoppelGANger [19] addresses overfitting and the risk of duplication of training data. Despite being crucial for tabular data generation assessment [2, 6, 12, 26], its importance remains underestimated for network flows.

Finally, Table 2 showcases the trend of recognizing generated data effectiveness through post-generation machine learning (ML) performance (*utility* metrics, similar to [29]). We think that using ML-based methods for the evaluation cast some doubts on what is truly evaluated. Indeed, when evaluating the performance of a particular machine learning algorithm on a given task using both synthetic and real-world data, the outcomes are significantly impacted by the selected task and the chosen model, whereas we would want these outcomes to be only influenced by the discrepancies between the two datasets. While we concur with [53] on the necessity of à practical assessment for task-specific data, we thus claim that this kind of assessment is not sufficient for comprehensive and interpretable results.

3 METRICS FOR SYNTHETIC NETWORK TRAFFIC ASSESSMENT

In this section, we introduce the set of metrics that constitute our evaluation system. Our approach to evaluate synthetic data generation is characterized by its two fundamental attributes: comprehensiveness and granularity. Specifically, we ensure that our evaluation system

Evaluation method	Criteria				Descriptions		
	Realism	Diversity	Novelty	Compliance	Ref.	Input	Type
Network Response				✓	[8, 25]	Samples	Boolean
Byte Error Rate				✓	[8, 25]	Samples	Numerical
Marginal distributions distance	✓	✓			[7, 23, 34, 50]	Marg. Distr	Visual
Euclidian Distance	✓	✓			[34]	Marg. Distr.	Numerical
Domain Knowledge Check				✓	[34, 39, 50]	Samples	Numerical
Classifier performance loss	✓	✓			[7, 19, 46, 50, 51, 53]	Joint Distr.	Numerical
EMD	✓	✓			[7, 19, 23, 39, 47], [51]	Marg. Distr.	Numerical
JSD	✓	✓			[17, 50], [51]	Marg. Distr.	Numerical
False Negative Test	✓				[7, 39]	Samples	Boolean
PCA Analysis	✓	✓			[25, 47]	Joint Distr.	Visual
KS Test	✓	✓			[7, 10, 47]	Marg. Distr.	Numerical
Correlation Matrix Comparison	✓				[7, 19, 27]	Joint Distr.	Visual
Chi-Squared Test	✓	✓			[7]	Marg. Distr.	Numerical
Membership disclosure			✓		[19]	Sample	Numerical

Table 2: Summary of the functions used to evaluate generated network traffic. Marg. Distr.: Marginal Distribution, Joint Distr.: Joint Distribution

comprehensively addresses all four criteria presented in Subsection 2.3. Moreover, our evaluation framework is designed to offer a fine-grained analysis, allowing us to identify specific limitations of the benchmarked data generation methods.

To formalize our evaluation benchmarks, we outline the main requirements as follows:

- **Criterion-based Assessment**: the evaluation process must encompass benchmarks that pertain to the four predefined criteria: *Realism*, *Diversity*, *Novelty*, and *Compliance*.
- **Attribute-level and Sample-level Evaluation**: recognizing the unique characteristics of tabular data, we advocate evaluating *Realism* and *Diversity* both at the attribute level (marginal distribution) and the sample level (joint distribution).
- **Data Type Specificity**: to address potential issues related to different data types (numerical/categorical), the evaluation protocol should incorporate metrics tailored to each data type.
- **Individual Criterion Assessment**: each of the four criteria should undergo individual assessment to ensure a comprehensive evaluation.

Drawing inspiration from these requirements, we propose a performance measurement system that involves:

- Comparison of the marginal, the conditional and the joint distributions to assess Realism and Diversity of the synthesized network traffic.
- Measuring membership disclosure to assess Novelty of the generated traffic.
- Conducting domain knowledge check to assess Compliance of the synthetic data to network protocols.

The rest of this section consecutively presents these aspects.

3.1 Marginal, conditional and joint distributions of the generated data

If two distributions have the same joint distribution, they have the same conditional and marginal distributions. In this regard, comparing the joint distributions of the generated data with the evaluation data is enough to evaluate the generation model. However, the joint distributions are difficult to compare due to the sparseness of the generated data. For this reason, and to ensure the metrics' granularity, we propose to also evaluate the marginal and conditional distributions.

3.1.1 Marginal distribution of the generated data. Building upon the insights from [7], we propose an evaluation strategy that segregates the assessment of numerical features from that of categorical features. This approach allows to identify any potential challenge a model faces in generating either continuous or discrete data types.

For our evaluation, we utilize *Jensen Shannon Divergence* (JSD) for discrete attributes of network traffic, such as *Protocol*; and *Earth Mover's Distance* (EMD) for continuous attributes like *Duration* or *Bytes*. We make this choice based on their established prevalence within the state of the art. Notably, these metrics offer a distance metric, which distinguishes them from statistical tests of independence like *Kolmogorov-Smirnov Test* or *Chi-squared Test*.

JSD quantifies the similarity between the probability mass functions of real and synthetic data for a given discrete feature. It is a symmetric variant of the Kullback-Leibler divergence. Importantly, JSD is calculated independently for each variable, thus focusing solely on individual variables without capturing inter-variable dependencies. Eq. 1 gives the formulation of JSD for a discrete feature.

$$JSD(\mathrm{X}_{\mathrm{source}} \,\|\, \mathrm{X}_{\mathrm{gen}}) = \frac{1}{2} D_{\mathrm{KL}}(X_{source} \,\|\, M) + \frac{1}{2} D_{\mathrm{KL}}(X_{gen} \,\|\, M) \quad (1)$$

where $M = \frac{1}{2}(\mathrm{X}_{\mathrm{source}} + \mathrm{X}_{\mathrm{gen}})$ and $D_{\mathrm{KL}}(P \,\|\, Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$.

EMD is calculated by analyzing the real and synthetic Cumulative Density Functions (CDFs) of a specific variable. This metric quantifies the *amount of mass* required to be displaced in order to transform the source CDF into the generated CDF. Eq. 2 gives the EMD between the value distributions of a continuous feature for source network flow data and generated network flow data.

$$\begin{aligned} EMD(\mathrm{X}_{\mathrm{source}}, \mathrm{X}_{\mathrm{gen}}) &= W_1(\mathrm{P}(\mathrm{X}_{\mathrm{source}} \leq x), \mathrm{P}(\mathrm{X}_{\mathrm{gen}} \leq x)) \\ &= \int \left| \mathrm{P}(\mathrm{X}_{\mathrm{source}} \leq x) - \mathrm{P}(\mathrm{X}_{\mathrm{gen}} \leq x) \right| \mathrm{d}x \end{aligned} \quad (2)$$

3.1.2 Pairwise conditional distribution of the generated data. The aim is to assess whether the correlations observed between variables in real data are replicated in generated data. Common correlation metrics like Spearman or Pearson coefficients are typically applicable to ordered features. However, certain attributes, such as the

Protocol attribute, lack a natural order. Therefore, two distinct metrics are required: one to evaluate the correlation between numerical and ordered features and another to assess dependencies between categorical and unordered features.

For evaluating correlations among numerical features, we can analyze how their ordering correlates using the Spearman correlation coefficient, or we can explore the existence of linear relationships among them using the Pearson correlation coefficient. We believe that focusing solely on the preservation of ordering relationships is insufficient, especially for the type of numerical features under consideration. For instance, if the duration of a flow increases, the number of packets should also likely increase; however, if both real and generated data preserve such a property, it would be inadequate to conclude on the preservation of dependency between the two variables. This is why we chose to utilize the Pearson correlation coefficient.

Pairwise conditional distribution (PCD) [7, 12, 27] is the L^2 norm of the difference between two correlations matrices, as in Eq. 3:

$$PCD(\mathrm{S},\mathrm{G}) = \|\mathrm{Corr}(S) - \mathrm{Corr}(G)\|_2 \tag{3}$$

where $S = (X_{\mathrm{source}_i})_i$ is the set of numerical features of the source data, $G = (X_{\mathrm{gen}_i})_i$ is the set of numerical features of the generated data, and $\mathrm{Corr}(.)$ is the correlation matrix with Pearson coefficients.

CMD, as described in Eq. 4, is the disparity of the contingency matrices of a pair of features on the synthetic data or on the real dataset:

$$CMD(S,G) = \sum_{(i,j)\in[\![1,n]\!]^2} \Big(\sum_v \sum_u \mathrm{P}(\mathrm{X}_{source_i} = u | \mathrm{X}_{source_j} = v) - \mathrm{P}(\mathrm{X}_{gen_i} = u | \mathrm{X}_{gen_j} = v)\Big) \tag{4}$$

where $S = (\mathrm{X}_{source_i})_{i\in[\![1,n]\!]}$ is the set of categorical real features, and $G = (\mathrm{X}_{gen_i})_{i\in[\![1,n]\!]}$ is the set of categorical generated features.

3.1.3 Joint distribution of the generated data. The metrics based on PCD and CMD only consider first-order dependencies. However, in the context of network traffic data, many features depend on more than one feature. For example, the *Bytes* feature depends on both the number of packets and the protocol type. Therefore, to capture higher-level dependencies, it is crucial to assess the joint distribution.

There are two methods in the literature for evaluating the joint distribution: comparing the generated and real data distribution manifolds or using the difference in the performance of a machine learning (ML) algorithm on a classification task as a proxy for measuring the similarity between the two distributions.

As discussed in Subsection 2.3, using ML as a proxy is problematic as it heavily depends on the classification task and the ML method selected. Therefore, we opt to compare real and generated distribution manifolds. To achieve this, we believe that specific functions for both criteria are necessary to enforce the granularity requirement, thereby excluding methods like LogCluster or MSID (see Table 3).

Realism was traditionally assessed with Precision (see Table 1), but Naeem et al. [28] identified issues with this metric and proposed an improvement called Density, which we subsequently use. To compute Density [28], for each synthetic sample, we count how many real-sample neighborhood spheres contain the synthetic sample. The neighborhood spheres are computed using k-nearest neighbors. A low score indicates that real and synthetic data are far from each other, implying that the synthetic data is unrealistic.

Similarly, for *Diversity* Naeem et al. [28] proposed Coverage. To compute Coverage, for each real sample, we count the number of synthetic neighborhood spheres that include this sample. A low score suggests that several real samples lack a synthetic counterpart in their vicinity, indicating that the synthetic distribution does not adequately capture the variance of the real distribution.

Coverage and Density both rely on a specified number k of neighbors. Thankfully, [28] led an hyper-parameter optimisation. According to their results, when we consider a generated dataset and a real dataset of both 10000 samples, the optimal k should be set to 5. We therefore use this value for k and this number of samples in our comparative study.

3.2 Membership Disclosure

As indicated in Tables 1 and 2, three scores allow us to measure the *Novelty* of generated data. We chose to adapt the *Membership Disclosure* (MD) score from [12]. This score has been previously employed for network traffic [19] and shows a strong correlation with the Data Copying test, as illustrated in Appendix E. Moreover, it considers not only direct copies but also generated samples that closely resemble training instances. The objective of MD is to identify synthetic samples exhibiting characteristics that suggest they were copied from training instances.

To compute the MD score, we need a generated set, a training set, and a testing set (the last two sets being subsets of the real dataset). We begin by calculating the matrix of Hamming distances between every pair of generated and real samples. If a synthetic sample has a Hamming distance to a real sample below a certain threshold r, we flag the corresponding real sample as a leaked trained sample. Since we know which real samples are part of the training set or the testing set, for each r, we obtain a detector of training samples. Consequently, we can calculate the F1-score of such a detector and compute the integral of that F1-score depending on r. If the generated data includes instances copied from the training set, these copied samples would be detected even with a low threshold r, leading to an increase in the classifier's F_1 integral.

The interpretation of the value of this metric differs between the medical and network contexts. Medical data, being highly sensitive, necessitates synthetic data generation that respects the privacy of the patients whose data is used for learning. Thus, the MD score should be as low as possible. On the other hand, in a network context, encountering duplicated network flows, such as common DNS or NTP requests, is normal. Thus, we argue that Novelty in synthetic data should be close to Novelty observed in real data: synthetic data with a low MD score fails to capture the inherent duplication of network data.

3.3 Domain Knowledge Check

To evaluate the *Compliance* of the generated network traffic, we opt to adapt the Domain Knowledge Check (DKC) proposed in [34] to our context, as it is the sole measure suitable for assessing the conformity of network flows. Other Compliance metrics (see Table 2), such as Byte Error Rate and Network Response, were designed to evaluate Compliance at the packet level.

Our approach to assess *Compliance* involves utilizing a set of Boolean tests to examine the adherence of generated network flows

to standard network properties. It is essential to emphasize that the set of tests is dataset-dependent and should be customized accordingly for each dataset. Further details about the specific tests we employ can be found in Appendix D.

4 BAYESIAN NETWORK-BASED NETWORK TRAFFIC GENERATION

Generating network flows using GAN encounters several challenges. *Firstly*, numerous numerical features exhibit a multi-modal distribution of network traffic patterns. GANs are susceptible to mode collapse, where the generated network traffic data becomes overly focused on a single mode of the diverse patterns, resulting in the generator outputting only a subset of the original distribution. *Secondly*, certain categorical features of network flows entail a substantial number of categories, such as *Source Port*, which spans any number between 0 and 65535, leading to high-dimensional and sparse data. Given that GANs rely on a discriminator to discern between generated and real examples, this sparsity prompts discriminators that prioritize sparseness over realness [49], culminating in suboptimal generation. *Finally*, network flows encompass highly correlated features, such as *Protocol* and *Destination Port*, due to the regulation of network protocols, hardware architectures, and user behavioral patterns. GANs fail to explicitly capture these intrinsic correlations between different network attributes [29].

GAN-based approaches prioritize producing synthetic samples with a similar distribution to that of the real ones, instead of learning to preserve the correlations across different features.

Despite adaptations of GANs for network data generation, anomalies persist in the generated synthetic network traffic data. For instance, in Table 2 of [23], multiple flows exhibit ephemeral ports for both the source and destination, and one flow displays multiple packets with a null duration, which should only occur when a single packet is transmitted. Similar anomalies can also be found in the results of the most recent GAN-based method, NetShare [51], as exposed in Appendix C.

Contrary to GAN, Bayesian Networks (BN) inherently excel in representing the relational structures characteristic of tabular data. BN explicitly estimates the probabilistic dependency between data attributes, which reflects the intrinsic correlation embedded in the data source. This was previously highlighted in [5, 12]. Particularly, Goncalves et al. [12] extensively compared various GAN and traditional statistical methods for generating medical patient data. Their findings demonstrate that BN are better at generating medical data than GAN across diverse datasets, as indicated by multiple evaluation metrics. Besides, BN has a much lower training cost than GAN, which typically requires expensive hardware and many hyperparameter tweaks to obtain a satisfying convergence. Finally, BN has the key advantage of the ability to facilitate probabilistic reasoning by learning dependencies from data.

In our study, we thus propose to use BN to generate synthetic network traffic data, in the form of NetFlow data structure. Our conjecture is two-fold. For one thing, network traffic data are highly structured and constrained by the underlying network protocols and the usage pattern behind the network activities. It induces strong correlation between the features of NetFlow data. For the other thing, BN based method is good at depicting the conditional dependency between network features. Benefited from the characteristic of BN, it is intrinsically suitable for generating synthetic network traffic data. In the followings, we describe the details about the BN-based data generation method.

Structure Learning Method	BIC score (Higher is better)
Naive Bayes	-1.94×10^7
Chow-Liu	-1.80×10^7
Hill climbing	-1.68×10^7

Table 3: Comparison of the different structure learning algorithms of the library bnlearn library in Python on the UGR'16 Dataset.

Bayesian networks [31] are composed of a directed acyclic graph, where each node N is labelled by one feature X_N and is associated with a conditional probability table that describes the distribution $P(X_N \mid X_{Pa(N)})$ where $X_{Pa(N)}$ are the features of the parents of N. Hence, the graph represents dependencies between variables. BN can represent any probability distribution due to Bayes' theorem:

$$P(X) = \prod_N P(X_N \mid X_{Pa(N)}) \tag{5}$$

There are multiple families of BN learning algorithms: constraints-based ones, which build the network of dependencies with statistical tests; score-based ones, which seek to maximize the score by modifying the network iteratively; and hybrid methods. For choosing the type of structure learning, we simply try to fit different algorithms of the bnlearn Python library [1] on the UGR'16 dataset and computing their respective BIC score (see Table 3). You can also find an example of structure learned during this experiment in Apprendix A.

As we can see on Table 3, Hill-Climbing is the structure algorithm that get the best score on that small experiment, we therefore decide to use it to construct our solution.

Given a collection of data points $\Omega = (x_i)_{i \leq n}$, a model θ, and k the number of parameters in θ, BIC is defined by Eq.6:

$$BIC(\theta \mid \Omega) = -2\sum_{i=1}^{n} \log(\mathrm{P}(x_i|\theta)) + k\log(n) \tag{6}$$

Once a Bayesian network model is learned, it is easy to generate data according to the probability distribution described by the model: by iterating over the nodes in a topological order, one can sequentially sample feature values according to the conditional probability table given the value of the parent nodes.

While Bayesian networks can handle numerical features, they typically work better with categorical features. Applying BN to network flow generation therefore requires discretizing the numerical features of network flows, e.g., the duration of a flow. To do so, we experiment with two discretization strategies. The first strategy, called $\mathbf{BN_{bins}}$, regroup values in quantiles. The second strategy ($\mathbf{BN_{GM}}$) leverages Gaussian Mixture Model (GMM), and more specifically Variational Gaussian Mixture [3], to fit the marginal distribution of the numerical features. This approach is particularly apt to network flows generation since numerical features in network configurations often exhibit multimodal marginal distributions. Thus, we map a numerical value to the index of its nearest Gaussian kernel. Both strategies are experimentally compared in Section 5.

[1] https://github.com/erdogant/bnlearn/

Another limit of BN is linked to the size of the model. The size of the discrete conditional probability tables is $O(d^{k+1})$ where d is the highest cardinality of the features and k is the maximum number of parents in the directed acyclic graph. Hence, the cardinality of the features has a huge impact on the size of the model. However some features of network flows have a high cardinality, such as IP addresses (4 billion values) and ports (65536 values). It is necessary to reduce the cardinality of those features.

While port value has a range from 0 to 65536, most values hold negligible information regarding communication content. This is especially relevant for ephemeral ports. These ports are dynamically assigned upon communication establishment and remain valid throughout the communication duration. To reduce the cardinality, we assign a single value to all ephemeral port instances. Since the range of ephemeral ports depends on the operating system, we propose retaining the 30 most frequent port values as application ports, categorizing the remaining ports as ephemeral ports. We choose 30 by studying the distribution of port values in UGR'16 and CIC-IDS2017.

IP addresses can be divided into public and private categories. For NIDS evaluation, private IP addresses hold intrinsic significance as they denote internal hosts requiring protection. Conversely, public IP ranges primarily designate external hosts and might not convey independent information, often anonymized in the NIDS community [22, 35, 40]. To reduce the cardinality of IP values, we choose to assign a single value to all public IP addresses.

Several features are thus modified: discrete features with a large cardinality and numerical features. Hence, generated values must be transformed back into the original domain of the feature. Ephemeral ports and public IPs are uniformly drawn among the list of ephemeral ports and public IPs in the original dataset. For numerical features, if the first strategy (based on bins) is used, the values are drawn uniformly from the bin's range. If the second strategy (based on GMM) is used, the value is sampled from the Gaussian kernels.

5 EXPERIMENTS

This section is dedicated to the experimental evaluation of several generation methods. We first describe our experimental protocol and then present our results.

5.1 Experimental protocol

To comprehensively evaluate the generation of synthetic network flows, we use seven approaches:

- The two variants of the proposed BN-based network traffic generation method. These variants are both built as outlined in Section 4; they only differ for the data discretization preprocessing stage:
  - $\mathbf{BN_{GM}}$: in this BN model, the discretization process is based on a Gaussian Mixture model.
  - $\mathbf{BN_{bins}}$: in this BN model, the discretization process divides the range of n numerical features into quantiles using bins. For each continuous feature, the range is split into 40 quantiles.
- three GAN-based approaches:
  - **E-WGAN-GP** [34]: this method holds significance as one of the foundational work in the realm of GAN-based traffic generation.
  - **NetShare [51]**: NetShare is a successor to DoppelGANger [19], which introduced the concept of utilizing a sequence generator for network traffic flow data.
  - **CTGAN [49]**: despite being designed to generate general tabular data, **CTGAN** showcases the performance expected from a GAN when a more generic solution is applied to network flow generation [7, 29].
- a naive approach (called **Naive Sampler** or **Naive** in the rest of the article). This **Naive Sampler** allows to assess the need for more complex and costly methods. To generate data, the Naive Sampler simply draws, for each attribute, a value from the training set. The sampling process of each attribute is independent. Consequently, the generated data are not realistic nor compliant to network protocols. However, we may expect them to have a high *Novelty* level.
- and, finally, real data different from the one used in training, but captured in the same environment and therefore with the same characteristics.

The comparative test between these seven approaches is organized using the evaluation metrics we propose in Section 3 and based on three datasets with a different number of features:

- We build two datasets from CIC-IDS2017 [40], by an updated version of CICFlowMeter proposed by [18] extracting unidirectionnal feature. The first dataset is named **CICSmallFeatureSet**. It is characterized by a limited number of network features (11 features). In contrast, the second dataset, namely **CICLongFeatureSet**, encompasses a larger feature set (30 features). By comparing the results on **CICSmallFeatureSet** and **CICLongFeatureSet**, we have two objectives. First, we aim to verify the impact of feature dimensions over the quality of generated network flows. Second, as **CICLongFeatureSet** embodies a more intricate dataset characterized by a greater number of numerical features (as shown in Appendix B), and this inherent complexity poses a greater challenge for BN. We aim to assess the impact of the number of numerical features on the resultant quality of the generation.
- The third dataset is from UGR'16 [22], which contains real-world network traffics recorded by a Spanish ISP. We choose to work with a subset that has been preprocessed by NetShare authors and range within the third week of March 2016 [51]. We name this subset as **UGR** in our study. This dataset consists in real traffic with 8 features per flow. We choose this specific subset to facilitate a more equitable and meaningful comparison with NetShare. Making comparison over that dataset will show how the data generation methods perform with real-world traffic, compared to testbed-produced simulations.

Before using these three datasets (textbfCICSmallFeatureSet, textbfCICLongFeatureSet and textbfUGR), we apply to them the data preprocessing technique proposed in Section 4. Therefore, we restrict the possible values of ports to the 30 most frequent ones: if a port value is not among them, we assign the arbitrary value *99999* to it. This value will serve for all port values that are not frequent enough in our dataset. Similarly, we apply the default IP *0.0.0.0* to all public IP addresses. This value is the default route, so it could be used to represent all external IP addresses. This is done for textbfCICSmallFeatureSet and textbfCICLongFeatureSet. Since textbfUGR is

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.017**	**0.017**	**0.025**	0.031	0.148	0.090	*0.23*
EMD	Realism and Diversity for numerical features (↓)	**0.002**	**0.002**	**0.014**	*0.080*	0.016	0.055	0.062
CMD	Realism of correlation between categorical features (↓)	**0.013**	0.160	**0.018**	**0.018**	0.126	**0.071**	*0.257*
PCD	Realism of correlation between numerical features (↓)	**0.761**	1.186	**0.630**	**0.891**	0.949	1.152	*1.191*
Density	Realism of data distribution (↑)	**1.000**	*0.079*	**0.906**	**0.887**	0.867	0.862	0.391
Coverage	Diversity of data distribution (↑)	**0.967**	*0.161*	**0.952**	**0.955**	0.903	0.655	0.214
MD	Novelty (=)	**7.175**	5.766	**6.929**	**6.987**	6.862	6.864	*5.247*
DKC	Compliance (↓)	**0.003**	*0.088*	0.004	**0.003**	0.008	**0.002**	0.023
Global Rank	Average Ranking (↓)	**1.2**	4.9	**2.2**	**2.9**	4.1	3.9	*6.0*

Table 4: Comparison, according to all our metrics, of 7 data generation methods using CICSmallFeatureSet. The Test columns serves as a standard. For each metric : *Red* indicates the worst model, **Orange** the second-best model and **Green** the best model. (↑): Higher is better, (↓): Lower is better, (=): Closest to the real data is better. The last line gives the average rank given by all metrics to each model, and is here just as an indication of overall performance.

solely composed of external IP flows, we decide not to consider the IP features for this dataset.

5.2 Experimental results

In this section, we successively present our results on the three datasets **CICSmallFeatureSet**, **CICLongFeatureSet**, and **UGR**. For each of them, we comment on the 4 criteria (*Realism*, *Diversity*, *Novelty*, and *Compliance*) and then give some elements on the computational cost. Finally, we conclude with some more general observations.

5.2.1 Results on CICSmallFeatureSet Data. We train the 7 network flow generation methods we consider on **CICSmallFeatureSet** and use these models to produce synthetic network flows. The network traffic generated by each model is evaluated according to the metrics presented in Section 3. The results are presented in Table 4.

Realism. With a first focus on the *Density* metric, which measures how closely the distribution of synthetic network flows data aligns with the real distribution, we see that the two BNs outperform the GAN-based methods.

To assess Realism in more depth, we can look at *CMD* and *PCD*: these metrics evaluate the extent to which generated data maintains statistical correlations among different attributes in training data. Lower *CMD* / *PCD* scores mean better preservation of intrinsic statistical correlations. As expected, the Naive Sampler performs poorly with *CMD* (0.160) and *PCD* (1.186) metrics due to its independent sampling of features. NetShare performs even worse in these correlation metrics (0.257 for *CMD* and 1.191 for *PCD*), struggling to capture underlying feature associations. By comparison, the other GAN-based approaches are shown to be better at preserving the correlation between features. E-WGAN-GP and CTGAN reaches a *CMD* score of 0.071 and 0.126 respectively, lower than those of the Baseline and NetShare. Similarly, E-WGAN-GP and CTGAN's *PCD* scores are 1.152 and 0.949. Both BN-based approaches show consistently lower *CMD* and *PCD* scores than the other approaches. That is expected because BNs are tailored to learn dependencies between variables.

A consistent ranking pattern emerges across *Density*, *CMD*, and *PCD* metrics for assessed data generation methods. The BN-based approaches consistently excel in preserving real data's distribution and correlations, while the GAN-based methods follow, outperforming NetShare and Naive Sampler in distribution preservation.

Diversity. *Coverage* gives us a first global view relatively to that criterion. For that metric, the two BNs outperform the other methods.

Extending our analysis to feature marginal distributions, we employ *JSD* and *EMD* to quantify Diversity in categorical and numerical features of network flows. As expected, Naive Sampler excels (0.017 for *JSD* and 0.002 for *EMD*), directly sampling real marginal distributions. BN_{GM}, while presenting a rather good *JSD* for discrete features (0.031), lag behind GAN-based approaches in capturing numerical feature distribution (*EMD* of 0.080). This may be due to the discretization process based on GMM, that may didn't learn correctly the marginal distribution of numerical feature. However, due to the low number of numerical feature in the dataset (3 out of 11), the impact on the Density of BN_{GM} generation is limited (*Coverage* of 0.955).

Novelty. *MD* is employed to detect that some generated samples are copies of source samples. A higher *MD* value suggests a greater likelihood of copied training data in the generated network flows. For the Real data, the *MD* is 7.175, indicating that some amount of copying should be expected in generated data. The BN-based methods have the closest MD value of the real data (6.929 for BN_{bins} and 6.987 for BN_{GM}). E-WGAN-GP and CTGAN seem less prone to data-copying. NetShare has the lowest *MD*: this is due to the generation of unrealistic samples, which therefore have a lower chance to be copied from the source data.

Compliance. We evaluate the Compliance by submitting the generated network flows of our different models to a series of tests defined in Appendix D. The *DKC* score is defined as the percentage number of failed tests over the whole set of Compliance tests. A higher *DKC* score, such as for NetShare (2.3%) and the Naive Sampler (8.8%), indicates that the generated traffic is less compliant to the network protocols. Except for NetShare, all models have a pretty low *DKC* score. We can also notice that the *DKC* score of the real data

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.018**	**0.018**	0.116	**0.078**	0.139	0.109	*0.359*
EMD	Realism and Diversity for numerical features (↓)	**0.001**	**0.001**	**0.014**	*0.087*	0.017	0.065	0.031
CMD	Realism of Correlation between categorical features (↓)	**0.010**	**0.079**	0.175	**0.032**	0.124	0.096	*0.384*
PCD	Realism of Correlation between numerical features (↓)	**2.345**	6.517	**2.761**	**3.456**	3.700	4.843	*8.404*
Density	Realism of data distribution (↑)	**0.997**	*0.051*	0.320	0.486	**0.647**	**0.492**	0.263
Coverage	Diversity of data distribution (↑)	**0.969**	*0.085*	0.647	**0.778**	**0.809**	0.416	0.120
MD	Novelty (=)	**7.612**	5.297	4.251	**5.663**	5.522	**5.835**	*2.539*
DKC	Compliance (↓)	**0.003**	*0.128*	0.060	**0.051**	**0.052**	0.092	0.055
Global Rank	Average Ranking (↓)	**1.0**	4.6	4.2	**3.1**	**3.5**	4.1	*5.9*

Table 5: Comparison, according to all our metrics, of 7 data generation methods using CICLongFeatureSetDat. The Test columns serves as a standard. For each metric : *Red* indicates the worst model, **Orange** the second best model and **Green** the best model. (↑): Higher is better, (↓): Lower is better, (=): Closest to the real data is better. The last line gave the average rank given by all metrics to each model, and is here just as indication of overall performance.

is not null. That means that some abnormal flows are present in the initial dataset.

Computational Cost. Table 7 shows the computing cost with the duration of three steps: the preprocessing phase, the time required to preprocess the training data (discretization for example); the training phase; and the sampling phase, denoting the time taken to synthesize artificial samples from the model. We also provide the hardware settings in our study. According to Table 7, the three fastest models in Experiment 1 are: BN_{bins}, BN_{GM}, and E-WGAN-GP. In particular, E-WGAN-GP is quick to train but its complex IP2Vec [33] embedding reconstruction process involves exhaustive searches for nearest neighbors, leading to a prolonged sampling duration. Furthermore, BN_{GM}'s discretization process necessitates training GMMs for each numerical feature, which is pretty long. On the contrary, BN_{bins} emerges as the fastest synthetic sample generation approach.

General Observation. This experiment using **CICSmallFeatureSet** data clearly shows that the BN-based methods are better than the GAN-based methods, both in terms of generation criteria and computational costs. In particular, we find that NetShare is generally worse than the Naive Sampler.

While Bayesian Networks might not consistently surpass GAN models across all metrics (e.g., *EMD*), their performance is notably strong, with an average ranking that positions them favorably (as indicated by the Global Rank in Table 4). The weaker performance of the GAN-based data generation methods on metrics like *CMD* and *PCD* (metrics assessing the preservation of the dependencies) suggests that preserving correlation among variables remains a persistent challenge for the GAN-based methods. This is despite the tailored adaptations these models have introduced to address this issue. BN_{GM} requires more computational overheads than BN_{bins} due to the GMM-based quantization operation. Moreover, BN_{bins} tend to produce more realistic samples than BN_{GM}, for less computation cost.

5.2.2 Results on CICLongFeatureSet Data. This experiment aims to illustrate how the data generation methods behave with more network features in the training dataset. *CICLongFeatureSet* data has 20 more numerical features than *CICSmallFeatureSet* (see Appendix B). This is expected to bring challenges to the involved data generation methods, including the proposed BN-based data generation methods. The results are shown in Table 5.

Realism. The *Density* score shows that CTGAN and E-WGAN-GP learned well the joint distribution of the training data point. Looking at *PCD* and *CMD*, we can see that the correlation between features are well learned by the BN methods and CTGAN. Surprisingly, the Naive Sampler reproduced well correlation among discrete features, as indicated by its *CMD* of 0.079. For marginal distribution, *JSD* shows us that the BN methods did learn the marginal distribution better than the other methods. *EMD* show that BN_{GM} did not handle the numerical features, surely due to its discretization (0.087, the highest of all the models).

Diversity. BN_{GM} and CTGAN have the highest Coverage score (0.778 and 0.809 respectively). They managed to capture the important variance of the data distribution. Other methods are not as good, especially NetShare and the Naive Sampler (0.022 and 0.014 respectively).

Novelty. As per the *MD* metric, all the involved models produce synthetic samples that are not direct clones of training data. Compared to the previous experiment with **CICSmallFeatureSet**, which contains less features to learn, our different models introduce little novelty when compared to the real data.

Compliance. The increased number of numerical features does not bring any impact to the *DKC* metric. This shows producing unrealistic values for the features do not hinder the compliance of the data, as discussed in Appendix D.

Computational Cost. Due to the higher number of numerical features, the preprocessing step of BN_{GM} increased greatly. Similarly, the training time of all the GAN models is increased: from around 50% for CTGAN to up to 300% for E-WGAN-GP. In contrast, BN_{bins} still remains the fastest model to generate new synthetic samples.

General Observation. We see in this experiment on **CICLongFeatureSet** Data that BN_{GM}, although learning correctly the discrete features, produced less realistic samples compared to CTGAN. This

	Description	Real data	Naive	BN_{bins}	BN_{GM}	CTGAN	E-WGAN-GP	NetShare
JSD	Realism and Diversity for categorical features (↓)	**0.067**	**0.0068**	**0.066**	0.070	0.218	0.105	*0.399*
EMD	Realism and Diversity for numerical features (↓)	**0.002**	**0.002**	0.018	0.007	*0.029*	*0.029*	**0.003**
CMD	Realism of Correlation between categorical features (↓)	**0.037**	0.223	**0.031**	**0.040**	0.209	0.050	*0.578*
PCD	Realism of Correlation between numerical features (↓)	**0.373**	*1.222*	**0.452**	0.738	0.863	1.219	**0.542**
Density	Realism of data distribution (↑)	**0.951**	0.355	0.701	**0.855**	0.486	**0.702**	*0.027*
Coverage	Diversity of data distribution (↑)	**1.000**	0.805	0.792	**0.998**	0.802	**0.996**	*0.076*
MD	Novelty (=)	**8.692**	7.519	8.312	**8.316**	7.447	**8.341**	*5.675*
DKC	Compliance (↓)	**0.006**	0.079	**0.005**	**0.005**	0.019	**0.004**	*0.129*
Global Rank	Average Ranking (↓)	**1.6**	4.4	**3.1**	**2.9**	5.1	3.6	*5.8*

Table 6: Comparison, according to all our metrics, of 7 data generation methods using UGR Data. The Test columns serves as a standard. For each metric : *Red* indicates the worst model, **Orange** the second-best model and **Green** the best model. (↑): Higher is better, (↓): Lower is better, (=): Closest to the real data is better. The last line gave the average rank given by all metrics to each model, and is here just as an indication of overall performance.

is due to the high number of numerical features, and the difficulty of BN_{GM} with these. However, we also see with NetShare that generating unrealistic samples do not hinder Diversity or Compliance. This experiment illustrated the disjunction between the four criteria of the evaluation, and the necessity to assess them individually. In a same way, we see that, when we compare BN_{GM} and CTGAN, assessing Realism and Diversity only on marginal and conditional distributions (with *JSD*, *EMD*, *CMD* and *PCD*) is not sufficient for assessing Realism and Diversity globally. BN_{GM} overcomes CTGAN on the majority of those metrics, and yet is worst than CTGAN when it comes to modelling joint distribution, certainly because of the prevalence of numerical features and the difficulty of this model to learn the distribution of such features (it has the worst *EMD* of all models).

In general, we find that BN_{GM} performs better on a higher number of metrics, so Bayesian Networks remain on top in this experiment. CTGAN has also good results in this experiment, but at a really high computational cost. BN_{GM} creates higher quality samples while being cheaper to train.

5.2.3 Results on UGR Data. In the two previous experiments, the training dataset is composed of traffic generated inside the same physical testbed. By using a real-world dataset, we want to observe if the evaluation results of the synthetic traffic are consistent with real traffic as the training data. The results are presented in Table 6

Realism. Aligned with the experiment on **CICSmallFeatureSet**, the BN-based methods exhibit high Realism in *Density*, *CMD* and *PCD* scores. Among GAN approaches, NetShare reaches good results in *EMD* (0.003), while E-WGAN-GP manages to achieve good global Realism on *Density* (0.702).

Diversity. Globally, BN-based methods are better than the GAN-based methods in learning the marginal distribution of features, as we can see with *JSD* and *EMD*. The *Coverage* metric ranks E-WGAN-GP first, and NetShare last. Most of the evaluated models (apart from NetShare) produced diverse samples, with BN_{GM} on top. Same as for the first experiment, this might be due to the simpler distribution, which is therefore easier to cover. It can be worth to note that CTGAN, while having a good *Coverage* (0.802), have a rather low *Density* (0.486). This induces that while managing to cover pretty much all of the real distribution, CTGAN does create unrealistic samples: a phenomenon known as *mode invention*. A more extensive illustration of this phenomenon is given in Appendix F.

Novelty. Due to the simpler distribution (fewer features, and smaller cardinality per discrete feature) in the UGR data, it is easier for a data generation model to produce synthetic data close to a training data sample. As a result, the *MD* scores of all models except NetShare are close to the real data's *MD*. This experiment shows that the complexity of the dataset has an influence on the tendencies for generative models to introduce novelty: the more simple the dataset is, the less likely the generated model will introduce novelty.

Compliance. All the data generation methods, except NetShare and Naive Sampler, have a good *DKC* score. It shows that they are able to produce traffic data compliant to network protocols. NetShare fails to generate valid traffic, due to its inability to encode correlation between numerical features.

Computational Cost. Referring to Table 7, BN_{bins} emerges as the swiftest model. Notably, we abstained from training NetShare, thus relying on details from the authors' paper [51]. While E-WGAN-GP demonstrates overall swiftness compared to BN_{GM}, the remaining GAN-based methods exhibit notably slower training speeds.

General Observation. In summary, the BN-based approaches preserve better Realism, Diversity and Compliance in the generated network data compared to the other methods. Besides, the BN-based methods are less computationally intensive than the GAN-based ones, since the feature dimension of UGR data is smaller than that of **CICLongFeatureSet** data.

By evaluating each criteria independently, we also illustrate that CTGAN suffers from the issue of mode invention, a phenomenon that we further illustrate in Appendix F. This is a common problem for GAN-based generative methods.

NetShare performs poorly on this experiment also, even if the comparison was made with data directly given by the authors.

Model	Duration										
	1: CICShortFeatureSet			2: CICLongFeatureSet			3: UGR				
	Prep.	Train.	Samp.	Prep.	Train.	Samp.	Prep.	Train.	Samp.	**Epochs**	**Hardware**
BN_{GM}	00:22	00:36	-	1:48	00:44	-	00:19	00:32	-	-	Laptop CPU / 32GB RAM
BN_{bins}	-	00:39	-	-	00:45	-	-	00:32	-	-	Laptop CPU / 32GB RAM
E-WGAN-GP	-	00:11	00:46	-	00:35	0:55	-	00:20	0:21	100	Laptop CPU / 32GB RAM
CTGAN	-	15:02	-	-	19:30	-	-	13:27	-	300	A40 GPU / 48GB VRAM
NetShare	-	20:42	00:39	-	27:31	00:56	-	($\simeq$ 100 : 00)	-	100	A40 GPU / 48GB VRAM

Table 7: Computing costs of the three experiments. The format is hours:minutes. Prep.: Preprocessing of the data, Train.: Training of the model, Samp.: Sampling from the model. The training of NetShare on UGR is the order of magnitude given by the authors in their paper. A cell with "-" denotes that either this step does not occur or that it is so short that we consider its time to be negligible

6 KEY TAKEAWAYS

A comprehensive and interpretable evaluation system. In Section 3, we introduce an evaluation framework that is **comprehensive**, as it assesses all four criteria: *Realism, Diversity, Novelty, and Compliance*. Furthermore, this framework is **granular**, as it has the capacity to pinpoint limitations within the benchmarked data. Subsequently, we outline the inaugural comprehensive evaluation system designed to meet these criteria. This system encompasses 8 distinct metrics, forming the basis for the assessment of network traffic flow generation. Across varying datasets, the consistent empirical observations underscore the necessity of simultaneous evaluation of synthetic traffic using multiple criteria, thus avoiding favoritism towards specific metrics. For instance, as highlighted in Table 4, E-WGAN-GP boasts the lowest *DKC* score among all methods on **CICShortFeatureSet**. Nevertheless, it was worst than our BN-based methods when we look to the Realism and Diversity of the produced generated data, as evident from *JSD*, *EMD* and *PCD* scores. The pivotal takeaway is that concentrating on a solitary criterion or only a fraction thereof can potentially lead to erroneous choices for end-users among various network flow generation methods.

GAN does not show significantly better performances compared to simple models like BN for synthesizing network traffic. In Section 4, we present some of the problems that GAN-based methods face when generating network traffic flows. Subsequently, we introduce a novel and streamlined approach to generate network flows using Bayesian Networks, detailed in the same section. Inspired by the insights from [12], we proceed to execute a series of experiments in Section 5 to compare two BN-based methods with GAN-based approaches that are explicitly tailored for generating network flows.

Our empirical findings lead us to the conclusion that, when contrasted with the straightforward Bayesian Networks, Generative Adversarial Networks fail to justify their heightened complexity and prolonged training duration. Notably, GAN consistently underperform across all our experiments. Although GAN benefit from the powerful function approximation capabilities of deep learning architectures, their success in computer vision does not consistently translate to tabular data generation, such as network flow or healthcare data generation. This is certainly due to the data distribution of this kind of data that contains both discrete and numerical features. Therefore, the key takeaway from our study is to advocate for Bayesian Networks as the preferred solution for addressing the specific challenge of generating network flows.

7 CONCLUSION AND LIMITATION

This study introduces an extensive and highly interpretable evaluation system for synthetic network traffic, centered around four key criteria: Realism, Diversity, Novelty, and Compliance. This system facilitates a thorough assessment of the quality of synthesized network traffic data, allowing for an insightful interpretation of the strengths and limitations of AI-based network traffic synthesis methods. Additionally, we propose an innovative network flow generation method based on Bayesian Networks and compare it to state-of-the-art GAN-based approaches using our evaluation system. The comparative study spans three diverse and widely-used network traffic datasets.

According to our experimental findings, various GAN-based methods consistently exhibit lower effectiveness compared to our BN-based approaches in generating diverse synthetic network flows that closely resemble real traffic patterns. It is worth noting that a potential limitation of our study is the exclusion of temporal dependencies. Currently, we only generate Network Flow features without considering any temporal correlation. However, our hypothesis is that the proposed Bayesian Network-based network flow synthesis method can capture statistical correlations among different network attributes in the training dataset, aiding in encoding temporal dependencies between network flows. In the future, we plan to expand our study to explicitly address temporal correlations between successive network traffic records.

REFERENCES

[1] S. Abt and H. Baier. 2014. A Plea for Utilising Synthetic Data When Performing Machine Learning Based Cyber-Security Experiments. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop (AISec '14)*. Association for Computing Machinery, New York, NY, USA, 37–45. https://doi.org/10.1145/2666652.2666663

[2] A. Alaa, B. Van Breugel, E. S. Saveliev, and M. van der Schaar. 2022. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International Conference on Machine Learning*. PMLR, 290–306.

[3] D. M. Blei and M. I. Jordan. 2006. Variational inference for Dirichlet process mixtures. (2006).

[4] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. 2021. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[5] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[6] A. Borji. 2022. Pros and cons of GAN evaluation measures: New developments. *Computer Vision and Image Understanding* 215 (2022), 103329.

[7] S. Bourou, A. El Saer, T.-H. Velivassaki, A. Voulkidis, and T. Zahariadis. 2021. A review of tabular data synthesis using GANs on an IDS dataset. *Information* 12, 09 (2021), 375.

[8] A. Cheng. 2019. PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 0728–0734.

[9] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun. 2017. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *Proceedings of the 2nd Machine Learning for Healthcare Conference (Proceedings of Machine Learning Research)*, Finale Doshi-Velez, Jim Fackler, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens (Eds.), Vol. 68. PMLR, 286–305. https://proceedings.mlr.press/v68/choi17a.html

[10] G. Dlamini and M. Fahim. 2021. DGM: a data generative model to improve minority class presence in anomaly detection domain. *Neural Computing and Applications* 33 (2021), 13635–13646.

[11] B. Dowoo, Y. Jung, and C. Choi. 2019. PcapGAN: Packet Capture File Generator by Style-Based Generative Adversarial Networks. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 1149–1154. https://doi.org/10.1109/ICMLA.2019.00191

[12] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, and A. P. Sales. 2020. Generation and evaluation of synthetic patient data. *BMC medical research methodology* 20, 1 (2020), 1–40.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Networks. (2014). arXiv:stat.ML/1406.2661

[14] L. Han, Y. Sheng, and X. Zeng. 2019. A Packet-Length-Adjustable Attention Model Based on Bytes Embedding Using Flow-WGAN for Smart Cybersecurity. *IEEE Access* 7 (2019), 82913–82926. https://doi.org/10.1109/ACCESS.2019.2924492

[15] D. Heckerman. 2008. *A Tutorial on Learning With Bayesian Networks*. Vol. 156. 33–82. https://doi.org/10.1007/978-3-540-85066-3_3

[16] W. Hu and Y. Tan. 2023. Generating adversarial malware examples for black-box attacks based on GAN. In *Data Mining and Big Data: 7th International Conference, DMBD 2022, Beijing, China, November 21–24, 2022, Proceedings, Part II*. Springer, 409–423.

[17] S. Hui, H. Wang, Z. Wang, X. Yang, Z. Liu, D. Jin, and Y. Li. 2022. Knowledge Enhanced GAN for IoT Traffic Generation. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3336–3346. https://doi.org/10.1145/3485447.3511976

[18] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and E. Totel. 2023. Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes. In *Risks and Security of Internet and Systems*, Slim Kallel, Mohamed Jmaiel, Mohammad Zulkernine, Ahmed Hadj Kacem, Frédéric Cuppens, and Nora Cuppens (Eds.). Springer Nature Switzerland, Cham, 18–33.

[19] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. 2020. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 464–483. https://doi.org/10.1145/3419394.3423643

[20] Z. Lin, Y. Shi, and Z. Xue. 2022. IDSGAN: Generative Adversarial Networks For Attack Generation Against Intrusion Detection. In *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part III*. Springer-Verlag, Berlin, Heidelberg, 79–91. https://doi.org/10.1007/978-3-031-05981-0_7

[21] M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly. 2018. Are GANs Created Equal? A Large-Scale Study. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 698–707.

[22] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón. 2018. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security* 73 (2018), 411–424. https://doi.org/10.1016/j.cose.2017.11.004

[23] L. D. Manocchio, S. Layeghy, and M. Portmann. 2021. Flowgan-synthetic network flow generation using generative adversarial networks. In *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*. IEEE, 168–176.

[24] G. Marti. 2020. Corrgan: Sampling realistic financial correlation matrices using generative adversarial networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8459–8463.

[25] A. Meddahi, H. Drira, and A. Meddahi. 2021. SIP-GAN: Generative Adversarial Networks for SIP traffic generation. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. 1–6. https://doi.org/10.1109/ISNCC52172.2021.9615632

[26] C. Meehan, K. Chaudhuri, and S. Dasgupta. 2020. A non-parametric test to detect data-copying in generative models. (Apr 2020).

[27] S. Molnár, P. Megyesi, and G. Szabó. 2013. How to validate traffic generators?. In *2013 IEEE International Conference on Communications Workshops (ICC)*. 1340–1344. https://doi.org/10.1109/ICCW.2013.6649445

[28] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. 2020. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*. PMLR, 7176–7185.

[29] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proc. VLDB Endow.* 11, 10 (jun 2018), 1071–1083. https://doi.org/10.14778/3231751.3231757

[30] N. Patki, R. Wedge, and K. Veeramachaneni. 2016. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 399–410.

[31] J. Pearl. 2009. *Causality*. Cambridge university press.

[32] R. H. Randhawa, N. Aslam, M. Alauthman, and H. Rafiq. 2022. Evasion Generative Adversarial Network for Low Data Regimes. *IEEE Transactions on Artificial Intelligence* (2022).

[33] M. Ring, A. Dallmann, D. Landes, and A. Hotho. 2017. IP2Vec: Learning Similarities Between IP Addresses. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 657–666. https://doi.org/10.1109/ICDMW.2017.93

[34] M. Ring, D. Schlör, D. Landes, and A. Hotho. 2019. Flow-based network traffic generation using Generative Adversarial Networks. *Computers & Security* 82 (may 2019), 156–172. https://doi.org/10.1016/j.cose.2018.12.012

[35] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho. 2017. Creation of Flow-Based Data Sets for Intrusion Detection. *Journal of Information Warfare* 16 (12 2017), 41–.

[36] A. B. Sai, A. K. Mohankumar, and M. M. Khapra. 2022. A Survey of Evaluation Metrics Used for NLG Systems. *ACM Comput. Surv.* 55, 2, Article 26 (jan 2022), 39 pages. https://doi.org/10.1145/3485766

[37] M. Sarhan, S. Layeghy, and M. Portmann. 2022. Towards a standard feature set for network intrusion detection system datasets. *Mobile networks and applications* (2022), 1–14.

[38] A. Schoen, G. Blanc, P.-F. Gimenez, Y. Han, F. Majorczyk, and L. Mé. 2022. Towards generic quality assessment of synthetic traffic for evaluating intrusion detection systems. In *RESSI 2022-Rendez-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information*.

[39] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar. 2020. Generative Deep Learning for Internet of Things Network Traffic Generation. In *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. 70–79. https://doi.org/10.1109/PRDC50213.2020.00018

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *International Conference on Information Systems Security and Privacy*.

[41] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security* 31, 3 (2012), 357–374. https://doi.org/10.1016/j.cose.2011.12.012

[42] R. Shwartz-Ziv and A. Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90. https://doi.org/10.1016/j.inffus.2021.11.011

[43] Y. Song, S. J. Stolfo, and T. Jebara. 2011. Behavior-based network traffic synthesis. In *2011 IEEE International Conference on Technologies for Homeland Security (HST)*. IEEE, 338–344.

[44] L. Deriand NETikos SpA. 2003. nProbe: an open source netflow probe for gigabit networks. In *TERENA Networking Conference*. 1–4.

[45] A. Tiwari, S. Saraswat, U. Dixit, and S. Pandey. 2022. Refinements In Zeek Intrusion Detection System. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Vol. 1. IEEE, 974–979.

[46] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang. 2020. PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC40277.2020.9148946

[47] K. Wilailux and S. Ngamsuriyaroj. 2021. Novel Bi-directional Flow-based Traffic Generation Framework for IDS Evaluation and Exploratory Data Analysis. *Journal of Information Processing* 29 (01 2021), 256–265. https://doi.org/10.2197/ipsjjip.29.256

[48] M.-J. Woo, J. P. Reiter, A. Oganian, and A. F. Karr. 2009. Global measures of data utility for microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality* 1, 1 (2009).

[49] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. 2019. *Modeling Tabular Data Using Conditional GAN*. Curran Associates Inc., Red Hook, NY, USA.

[50] S. Xu, M. Marwah, M. Arlitt, and N. Ramakrishnan. 2021. STAN: Synthetic Network Traffic Generation with Generative Neural Models. In *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 3–29.

[51] Y. Yin, Z. Lin, M. Jin, G. Fanti, and V. Sekar. 2022. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 458–472.

[52] S. Zhou, S. Ermon, A. Y. Ng, M. S. Bernstein, and Stanford University. Computer Science Department. 2021. *On the Evaluation of Deep Generative Models*. Stanford University. https://books.google.fr/books?id=2qGDzgEACAAJ

[53] P. Zingo and A. Novocin. 2020. Can GAN-Generated Network Traffic be used to Train Traffic Anomaly Classifiers?. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 0540–0545. https://doi.org/10.1109/IEMCON51383.2020.9284901

[54] P. Zingo and A. Novocin. 2021. *Introducing the TSTR Metric to Improve Network Traffic GANs*. 643–650. https://doi.org/10.1007/978-3-030-73100-7_46

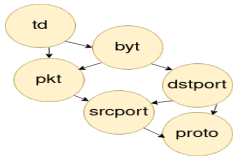

Figure 1: Example of Bayesian Network trained on UGR'16 with Hill-Climbing (see Section 4). Nodes are features, arrows are dependencies.

A LEARNED BAYESIAN NETWORK ON UGR'16

On Figure 1 we can see the Bayesian Network BN_bins that was trained on **UGR'16** with the Hill-Climbing structure learning algorithm. We see for example that the feature *byt* (corresponding to the number of Bytes sent) depends on the feature *td* (the duration of the connection). Those two features then determine the value of the feature *pkt*, the number of packet sent.

B DESCRIPTION OF THE FEATURES OF OUR DATASETS

We list in Tables 8 the different features used to describe network flows in all our experiments. In Experiment 1 and 2, the feature *Flags* only contains information about *URG* flags, *RST* flags or *PUSH* flags. Information about other TCP flags like *ACK*, *SYN* or *FIN* were unavailable in the dataset. We can see that in top of having the highest number of features, **CICLongFeatureSet** also has the highest number of numerical ones.

C EXAMPLE OF GENERATED NETWORK FLOWS

In Table 9 and Table 10 you can see the network flows generated by **BN_bins** and **NetShare** in the context of Experiment 1 (see 5.2.1).

As said in the previous subsection, the *Flags* feature comported only information on three types of TCP flags in the source dataset. So it is completely expected that none of our models produced any *ACK* flag or *SYN* flag for TCP flows. This is a shortcoming of the Dataset, not of our models.

We can see that **NetShare** traffic expose some serious shortcomings, like http traffic without any bytes in line 3, or flow with both *Dst Pt* and *Src Pt* as ephemeral ports at line 2.

D LIST OF TEST IN OUR COMPLIANCE EVALUATION

We will enumerate on Table 11 the list of network specification rules that we assess for our metric **DKC**. Those different rules are highly dependant on the feature set that we are working on, and thus they were not the same across our three experiments.

E HIGHLIGHTING OF DATA COPYING IN EXPERIMENT 3

We will show that Membership Disclosure MD, is indeed a good function to assess data-copying issue In order to do so, we asses if our generated data contains sample that are direct copy from training ones.

We do the comparison on our **UGR** dataset. The result are report on Table 12. We can see that our **MD** metric ranking match perfectly with this simple experiment. Our **MD** metric does not only account for samples that are exact copies, but also for generated samples that are really close to training samples without being copies. This shows us that our metric **MD** is an excellent proxy for evaluating data-copying.

Feature	Type	Dataset		
		Short	*Long*	*UGR*
Source IP Address	categorical	✓	✓	
Destination Port	categorical	✓	✓	✓
Destination IP Address	categorical	✓	✓	
Destination Port	categorical	✓	✓	✓
Protocol	categorical	✓	✓	✓
Timestamp	numerical		✓	✓
Day of the week	categorical	✓		
Hour of the day	numerical	✓		
Duration	numerical	✓	✓	✓
Number of packets	numerical	✓	✓	✓
Number of bytes	numerical	✓	✓	✓
Maximum length of a packet	numerical		✓	
Minimum lenght of a packet	numerical		✓	
Average length of a packet	numerical		✓	
Standard deviation of the length of packets	numerical		✓	
Sum of inter-arrival times	numerical		✓	
Average inter-arrival time	numerical		✓	
Standard deviation of the inter-arrival time	numerical		✓	
Maximum of the inter-arrival times	numerical		✓	
Minimum of the inter-arrival times	numerical		✓	
Flags inside the flow	categorical	✓		
Number of PUSH flags	numerical		✓	
Number of URGENT flags	numerical		✓	
Number of RESET flags	numerical		✓	
Sum of length of the headers	numerical		✓	
Average Number of Packets per second	numerical		✓	
Average of segment sizes	numerical		✓	
Average of Bytes/Bulk ratios	numerical		✓	
Average of Packets/Bulk ratios	numerical		✓	
Average of Bulk Rates	numerical		✓	
Number of packets inside a Subflow	numerical		✓	
Number of bytes inside a Subflow	numerical		✓	
Number of Bytes of the Init Window	numerical		✓	

Table 8: Description of the features of each flow in the three datasets of Subsection 5.1: *Short* stand for **CICShortFeatureSet**; *Long* for **CICLongFeatureSet**; and *UGR* for **UGR**

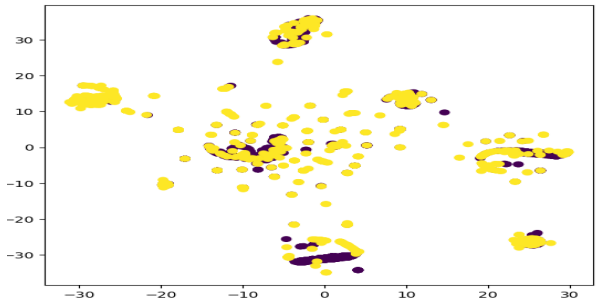

Figure 2: Two dimensional tSNE representation of UGR data and synthetic data generated by CTGAN. **Purple** is for UGR data and **Yellow** is for synthetic data

F ILLUSTRATION OF MODE INVENTION

In Subsection 5.2.3, when analyzing the result of CTGAN on the **UGR** dataset, we notice that its generation was lacking realism (*Density* of 0.486) yet had a pretty high Diversity (*Coverage* of 0.802).

	Day	Time	Duration	Proto	Src IP Addr	Src Pt	Dst IP Addr	Dst Pt	Packets	Bytes	Flags
1	4	18:10:02	0.08	UDP	192.168.10.8	51504	192.168.10.3	53	1	88	.....
2	3	17:17:02	5.66	UDP	192.168.10.8	49231	192.168.10.3	53	3	161	.....
3	0	19:56:16	59.36	TCP	192.168.10.19	41967	152.209.204.1	443	16	1409	..P...
4	2	19:47:39	115.85	TCP	192.168.10.15	46219	99.29.138.167	443	22	2758	..P...
5	4	13:28:26	0.03	UDP	192.168.10.1	53	192.168.10.3	61719	1	119	.....
6	4	13:38:46	119.01	TCP	192.168.10.5	53073	191.195.80.34	443	22	1235	..P...
7	1	19:35:24	0.07	UDP	192.168.10.3	53	192.168.10.17	58681	2	187	.....

Table 9: Example of network flows generated by BN_GM on Experiment 1.

	Day	Time	Duration	Proto	Src IP Addr	Src Pt	Dst IP Addr	Dst Pt	Packets	Bytes	Flags
1	2	16:45:42	0.02	UDP	192.168.10.5	49082	192.168.10.3	53	1	143	.....
2	1	12:12:14	2.49	TCP	157.69.249.80	51243	192.168.10.5	49653	5	1293	...R..
3	2	18:29:28	2.95	TCP	192.168.10.16	47793	87.64.65.152	80	3	0	.....
4	3	12:33:07	0.09	UDP	192.168.10.3	64087	192.168.10.1	53	1	316	.....
5	0	14:09:52	70.28	TCP	192.168.10.25	61487	192.168.10.3	3268	26	1963	..P...
6	4	16:25:51	0.05	UDP	192.168.10.1	53	192.168.10.3	53593	1	351	.....
7	0	17:09:37	6.06	TCP	192.168.10.8	49844	109.182.162.153	443	14	1199	..P...

Table 10: Example of network flows generated by NetShare on Experiment 1

Name	Feature involved	Experiments			Rule
		Short	Long	UGR	
Flags on TCP	Flags, Protocol	X	X		If the flow has flags, then Proto is TCP
At least one private IP	Source IP Address, Destination IP Address	X	X		One IP Address of the flow must be private
HTTP is over TCP	Destination Port, Source Port, Protocol	X	X	X	If one of a port is either 80, 443 or 80, it must be a TCP flow
DNS is over UDP	Destination Port, Source Port, Protocol	X	X	X	If one of a port is 53, it must be a UDP flow
No DNS reply to outside	Destination IP Address, Source Port,	X	X		If Source Port is 53, then the Destination IP Addresses should be private
No NETBios from or to outside	Source IP Address, Destination IP Address, Destination Port	X	X		If one IP Adress is public, then Destination Port should not ber 137/138
No HTTP reply to outside	Source Port, Destination IP Address	X	X		If Destintion IP Address is public, then Source port should not be 80/443/8080
Ephemeral Ports	Source Port, Destination Port	X	X	X	If one port is ephemeral, then the other should be an application port
No empty UDP or TCP flows	Protocol, Bytes	X	X	X	UDP or TCP flows should not have 0 byte
ICMP flows should be empty	Protocol, Bytes	X	X	X	An ICMP flow should have 0 bytes
No immediacy	Packets, Duration	X	X	X	If number of Packets greater than 1, Duration should be greater than 0
Sum of IAT cannot exceed Duration	Packets, IATs, Duration	X			The Duration should be superior to the sum of inter arrival time

Table 11: List of Tests carried out by the metric **DKC**, with the network rules they are assessing and the Experiment they are associated with. Short:Experiment 1-CICShortFeatureSet, Long:Experiment 2-CICLongFeatureSet, UGR:Experiment 3-UGR

	Test	Baseline	BN_bins	BN_GM	CTGAN	E-WGAN-GP	NetShare
MD (=)	**8.692**	7.519 (4)	8.312 (3)	**8.316 (2)**	7.447 (5)	**8.341 (1)**	*5.675 (6)*
% of Dt.Copying (=)	**66.27%**	2.73% (4)	17.24 (3)	**17.51% (2)**	1.67% (5)	**36.99% (1)**	*0.00% (6)*

Table 12: MD and Percentage of Data Copying in Experiment 3. Between parenthesis are the ranking of the generated set according to each method.

This behaviour is a known pitfall of generative model and is usually labelled as *mode invention*.

In order to visualize mode invention, we decide to embed both the training data and the synthetic data generated by CTGAN in a tSNE representation. We also plot that representation in a two dimensional graph in Figure 2.

We can see that every part of the real data distribution is pretty well covered by synthetic data (hence good Diversity), but the model did also produce a consequent amount of synthetic sample that are not close to any real one (the little independent dots in the center of Figure 2), indicating low Realism.