Generating Network Traffic Data Using Diffusion Model

Tong Shiqin, E0983364

Abstract
Generative models are playing an increasing role in various fields, and many products from the industry have shown that generative AI can significantly accelerate people's productivity in simple tasks such as generating copy, generating images, and so on. In addition, the role of generative AI in privacy protection is also attracting increasing attention, including but not limited to using generated fake data (consistent with the distribution of real data sources) to train AI models to protect the security of user data or reasoning and generating hard-to-access data, and then using this reasoning and generating data to deploy relevant models; Research has now shown that training with generated data not only achieves results that are almost similar to those of real data sets but also prevents problems that may arise in the future (e.g., preventing errors caused by incorrect samples that were not included in real data sets in the past, but will occur in the future). Based on all of the above, in this paper, we focus on the role of generative AI in privacy protection. Specifically, we focus on computer network protection: We use diffusion modeling to generate network traffic data containing both normal and attack events, and then we build downstream tasks and evaluation metrics to evaluate our generated data. The network data we generate can be used to identify network attacks instead of real data; thus, the privacy of computer networks is protected while ensuring accuracy. Experiments show that the data we generate performs similarly to real datasets in subsequent downstream tasks related to classification or recognition in artificial intelligence.

1 Introduction

How to use AI to generate practical data has been a focus of academic interest, and it has both commercial value and challenges. For commercial value, there are two aspects. First, the generated data can be substituted for real data and run in large distributed computing models. This allows for a massive increase in data privacy and solves the dilemma of data security and model accuracy. The second is that the absence and unevenness of data lead to uneven training and difficulty in improving accuracy. Meanwhile, for clean data in academia, in real life, we are often faced with incomplete data leading to a lack of model accuracy. For example: in the field of recognition, the deployment and utilization of the Yolo model is a mature industrial technology system, but while academics are trying their best to improve the accuracy of recognition of objects in the box, the problem faced by industry may be that the pictures are difficult to get the recognition box because of lighting and other factors. Therefore, generating clean data becomes the key to solving the problem of industrial model deployment when clean data is scarce in the industry. For the challenge of data generation, again, there are two aspects. The first is the diversity of data types, which is directly represented by the different feature dimension matrices of the data. Also, the relational mapping between data points can vary as a result. For example, an image dataset has completely different feature dimensions from textual sentence data. Therefore, even if the same generative adversarial framework based on the idea of confrontation is used for generation, the data preprocessing stage in processing these two types of data will be completely different, and the generative adversarial model will also have a lot of data-specific adjustments. The second is that the source and role of the data are different. Such differences can lead to differences in the background of the prior knowledge on which the data is based, and prior knowledge has been shown in many generative learning studies to have a large number of influences on the generation of data. Generative learning is a method of generating or fitting real data distributions from a Gaussian or other base distribution as a starting point, thus generating real data. A priori knowledge can determine the starting point, and the starting point significantly impacts the subsequent iterations to fit the real data.

In this paper, we focus on investigating the role of generative data in privacy computation in computer networks. Specifically, in computer security, the prevention of network attacks is an important topic, but there are two problems. The first one is the lack of datasets; such a lack is due to the private nature of network traffic data that makes some datasets not publicly available; the second one is that the accuracy of model training still needs to be improved; it is due to the fact that the proportion of attack data in the overall network traffic dataset is too small leading to the uneven samples in the dataset.

The use of generated network traffic datasets can solve the two problems mentioned above, thus allowing for a broader range of applications of AI models in computer security, including, but not limited to, the use of generated network traffic data to identify intrusions. This direction has also attracted a great deal of attention in the academic community, and there are currently two main categories of related research. The first category is to use machine learning methods to generate; this category currently includes the use of CNN, gan, and other methods to generate relevant traffic data and then use it to identify intrusion behavior. The second category uses many network simulators to generate relevant traffic data. This research category uses white-box (a priori knowledge methods) to generate and complement the dataset because of its good interpretability on the generated dataset. In this paper, we build on the first category of research and use machine learning methods to generate network traffic data; at the same time, we use downstream tasks such as identification or classification and relevant distributional similarity measures to determine the data we generate.

Unlike previous machine learning-based studies, we observe that the machine learning-based methods for generating network traffic data now use CNNs, GANs, etc... However, many studies have confirmed that GANs, which are the most widely used in generative learning, are less accurate than diffusion models. This does not mean that the diffusion model can fully replace the GAN, just as the triad theory proposed by xx; in terms of accuracy, speed, and trainability, no suitable model can take into account all three. The diffusion model is more robust than GAN regarding accuracy and trainability but needs to be fixed in speed. However, the traffic data of the network is not computed synchronously but asynchronously. Therefore, accuracy is more important than speed in the research of generating network traffic data. Therefore, it is of practical significance to try the diffusion model for generating network traffic data to get higher accuracy. At the same time, in the scope of our research, we found that there needs to be research on using the diffusion model in network security. Based on the above discussion, in this paper, we use the diffusion model to generate network traffic dataset and evaluate our generated data using downstream tasks and relevant evaluation metrics.

2 Background

In this section, we do a detailed literature review on generating network traffic data. We will briefly introduce the data types and describe the generative learning methods.

2.1 Data Types

Nowadays there are two types of network data; the first type is network packet data, which is in the form of pcap file which includes information about the network transport layer.The second category is network traffic data, which is in the form of tabular data, which includes information such as start ip address, end ip address, transmitted bytes, etc. Pcap files can be converted to tabular data. Regarding the data sources, the currently publicized data sources are as follows.

- UGR16: UGR16 is a real network traffic dataset developed by the Network Security Laboratory of the University of Granada (Spain). It supports cybersecurity research, such as network traffic classification and intrusion detection. The dataset contains traffic data from real network environments, including the communication of multiple applications and protocols such as HTTP, FTP, SMTP, SSH, etc. The UGR16 dataset is based on traffic data from IPv4 addresses and covers a wide range of network activities, including normal user behavior and malicious attacks.

- UGR17: This is the successor to UGR16.The UGR'17 dataset contains more data and has updated classification labels. It covers more applications and protocols to support network traffic classification and intrusion detection research.

- UNSW-NB15: This dataset is commonly used in intrusion detection research and for network traffic classification. It contains traffic data from lab networks covering various applications and protocols that can help with traffic classification and anomaly detection research.

- CIC-IDS2017: This dataset contains a mixture of network traffic from multiple applications and can be used to evaluate the performance of different algorithms and methods in classification and detection tasks.

- CTU-13: This is a dataset from the Czech Technical University (Czech Republic) for research on intrusion detection. It contains network traffic from 13 different Botnet attacks and is suitable for evaluating Botnet detection and intrusion detection algorithms.

In this paper, we use the UGR16 dataset.

2.2 Generative Learning for Network Traffic Data

2.2.1 Generating Adversarial Networks

GAN is based on the idea of a game and builds two networks, one as a generator and one as a discriminator. Its purpose is to generate a random free random distribution through the generator continuously; the discriminator is constantly modified to allow the generator to generate a highly similar distribution to the target in one step. In generating network traffic, there are also many studies using GAN to generate network traffic data. Research[5] combines the methods of oversampling a few (abnormal) classes and oversampling most (normal) classes to achieve better classifier performance. Research[12] uses GAN with rare class parallel learning to generate packets for DNS amplification attacks. Research[17] presents an application of deep learning methods to solve the imbalanced data problem in network traffic classification, and it uses auxiliary classifier generation adversarial networks to generate synthetic data samples. Research[18] proposes a novel traffic data enhancement method PacketCGAN using conditional

GAN. the PacketCGAN in Research[18] takes advantage of CGAN to generate specified samples conditional on the input application type to achieve data balancing. Research[14] proposes a new IoT traffic generator called IoTTGen. Research[2] proposes a systematic approach for identifying and quantifying the structure types of packet tracking features in communication networks and proposes a traffic generator model. The model can also generate synthetic traces that match the complexity level of their corresponding real-world traces. Research[10] uses trained generative models to generate spurious but plausible messages as test cases and proposes an automatic intelligent fuzzy testing framework (GANFuzz) for testing INP implementations. It can test several simulators of the Modbus-TCP protocol and find some bugs and vulnerabilities. Research[8] proposes Attack-GAN to generate adversarial network traffic at packet level that complies with domain constraints. Specifically, the adversarial packet generation is formulated into a sequential decision making process. In this case, each byte in a packet is regarded as a token in a sequence. The objective of the generator is to select a token to maximize its expected end reward.

2.2.2 Convolutional Neural Network

Convolution was first conceptualized in signal processing to deal with anomalous signals and was found to be used with great accuracy in images in the 2011 competition. Today, CNNs have become the underlying network for all kinds of AI models or frameworks. Thanks to the convolution of spatial translations, CNNs can characterize large amounts of Euclidean spatial data. Some studies in network generation use CNNs to generate network traffic. Due to the temporal nature of network traffic, many studies have also used LSTM to generate traffic. Meanwhile, Autoencoder is a neural network structure used to learn a low dimensional representation (encoding) of the input data and reconstruct the original input data from the encoding. Its goal is to capture the features and patterns of the input data by mapping the input data from a high-dimensional space to a low-dimensional space and reducing it again to the original input. These researchers have made some attempts. Research[7] proposes an alternative coding to encode network traffic data into CNN models. Experiments show that the traffic they generate can be successfully transmitted over the Internet to obtain the desired response from the network. Research[20] presents STAN (Synthetic network Traffic generation with Autoregressive Neural models) for generating realistic synthetic network traffic datasets. they use neural architecture(convolutional neural) captures both temporal dependencies and dependence between attributes at any given time and models both continuous and discrete variables. Research[9] uses LSTM networks to learn and generate packet sequences for less populated classes in the stream.[13] proposes a new architecture useing a Variational Autoencoder (VAE) and Recurrent Neural Network (RNN) to generate various kinds of network traffic while being able to do it in real-time. In Research[15], the autoencoder is trained to learn a latent representation of the real sequence of packet sizes. And the GAN is then trained on the latent space to learn to generate latent vectors that can be decoded as real sequences. Research[1] uses generative deep learning methods such as adversarial autoencoders (AAE) and bi-directional generative adversarial networks (BiGAN) to analyze network data. Research[6]proposes a differentially private autoencoder-based generative model (DP-AuGM) and a differentially private variational autoencoder-based generative model (DP-VaeGM).

2.2.3 Simulator

Simulation (emulation)[4] is a topic that has been discussed for a long time, and a large number of literature reviews survey emulation in networking. In general, network simulators do not generate network data. Instead, they simulate the behavior of a network based on a set of predefined parameters and inputs. The accuracy and validity of the simulation results depend on the quality and relevance of the input parameters and models used in the simulation. Such as Researches[3;11;16] describe the main properties that a network workload generator should have today, and present a tool for the generation of realistic network workload.

3 Method

Our study uses the diffusion model for network traffic data generation. Specifically, we use ugr16 data, build each step in the diffusion model, and then generate the relevant results. Before generation, we use some basic computational methods to fill in the missing values of the dataset, such as replacing NA values with 0 values; meanwhile, we perform standard feature processing.

3.1 Diffusion Model

The diffusion model is divided into two processes, the first process is the diffusion process, specifically, we have the original dataset, to which we add noise step by step according to the probability formula, through this process, we get the input (data after adding noise in the previous moment), the output (data with noise added at that moment), the noise at that moment for each moment. The diffusion process ends with the data being completely perturbed into a random free distribution. Then, we perform the second process. The first process reconstructs the original data from the random free distribution. Our goal is to input the distribution at one moment and the current moment, get the noise at that moment, then we subtract this noise to get the distribution at the next moment, then for the distribution at the next moment and the next moment t, we predict the noise at the next moment, then subtract the noise to get the distribution at the next moment. And so on until we can reconstruct the original data. The exact formula and key math is shown below:

$$\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

3.2 Code Details for Our Study

We illustrate, with simple code modules, the methods we use in each step of the diffusion modeling operation.

- Training Toolkit: we first need some helper functions for the network, these include log transformations, computation of likelihoods, and the use of one-hot representation of features.
- Forward Process: we build the functions for the forward process, a process that does not involve the learning of the network(purely a mathematical computation). We first define constants such as generating time steps and then add noise to each step until finally it becomes a completely random distribution. There is some interesting discussion

in current research regarding the number of time steps for diffusion model, but we use the common time steps in our study. Intuitively, the number of time steps for diffusion represents the size of the training set data at the second step of reconstruction, and there are some methods with training data in normal machine learning, such as too large training data can be overfitted whereas too small training data can lead to accuracy degradation (because the learned features are not sufficient to support the completion of the task). The discussion of the number of diffusion steps is closely related to the type of dataset for which the diffusion model is suitable.

- Reverse Recovery Process: the reverse process is needed to predict the noise at each moment. The input is the output of the previous moment. Therefore, we build the u-net as well as the transformer to predict the noise at each moment. Note that after obtaining the noise at each moment, we need to subtract the noise, and also there is a parameter z in the second part of the formula of the diffusion model, which controls the percentage of noise. In our study, we did not modify the magnitude of z, but followed the numerical magnitude set by z in the original diffusion model for the calculation. There are quite a few articles on z, most of which are explorations of the interpretability of z. The reason for using u-net is that we need to get the same input as the output. The reason for using transformer is that, in the data of network traffic, firstly, the temporal order is not negligible, and secondly, the traditional lstm has a memory cycle, specifically, the lstm only remembers a certain time range of information, and forgets the information of earlier time as the normal network in general. On the other hand, the transformer belongs to the information of the whole sentence, therefore, theoretically, the window of the transformer is infinite, and it can correlate the information of the whole sentence without the forgetting problem of the lstm. Meanwhile, since we are using tabular data, we need to modify the network of 2d to 1d and then perform the whole operation.

---

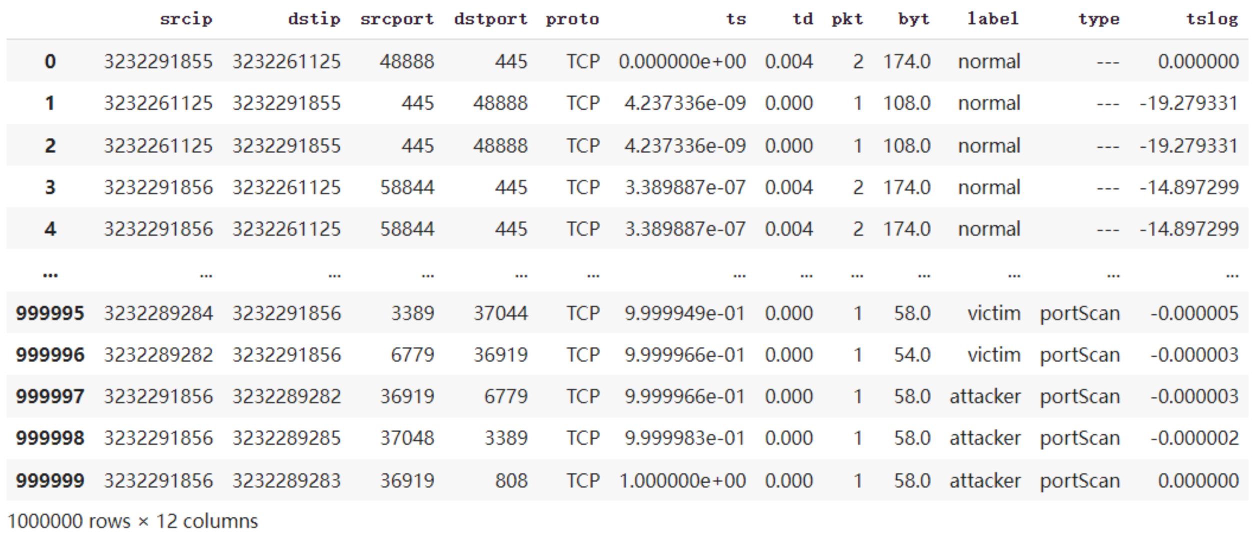
4 Experiment

4.1 DateSet

We use the network traffic datasets in UGR16 for our experiments. We counted the attack data and normal data in them. We found that the ratio of attack data to normal data in these used network traffic data is 1:2. Also, each dataset is 1 million. For each line of traffic data, we can see the source port, destination port, transmission start time, transmission end time, and the type of protocol used. These can be seen in Fig 1.

4.2 Replicate Works(Comparison Model)

- CTGAN (Conditional Table GAN)[19]is a powerful generative model designed specifically for generating synthetic tabular data. It builds upon the widely-used framework of Generative Adversarial Networks (GANs) to address the challenges posed by the unique characteristics of tabular data. One of the key features of CTGAN is its ability to handle both continuous and discrete features in the tabular data. It categorizes the tabular data into continuous and discrete data. For continuous data it uses hybrid Gaussian for a priori fitting and for discrete data it uses one-hot representation. Then the a priori optimized data is fed into the conditional GAN as a starting point for generation.



	srcip	dstip	srcport	dstport	proto	ts	td	pkt	byt	label	type	tslog
0	3232291855	3232261125	48888	445	TCP	0.000000e+00	0.004	2	174.0	normal	---	0.000000
1	3232261125	3232291855	445	48888	TCP	4.237336e-09	0.000	1	108.0	normal	---	-19.279331
2	3232261125	3232291855	445	48888	TCP	4.237336e-09	0.000	1	108.0	normal	---	-19.279331
3	3232291856	3232261125	58844	445	TCP	3.389887e-07	0.004	2	174.0	normal	---	-14.897299
4	3232291856	3232261125	58844	445	TCP	3.389887e-07	0.004	2	174.0	normal	---	-14.897299
...	...	...	...	...	...	...	...	...	...	...	...	...
999995	3232289284	3232291856	3389	37044	TCP	9.999949e-01	0.000	1	58.0	victim	portScan	-0.000005
999996	3232289282	3232291856	6779	36919	TCP	9.999966e-01	0.000	1	54.0	victim	portScan	-0.000003
999997	3232291856	3232289282	36919	6779	TCP	9.999966e-01	0.000	1	58.0	attacker	portScan	-0.000003
999998	3232291856	3232289285	37048	3389	TCP	9.999983e-01	0.000	1	58.0	attacker	portScan	-0.000002
999999	3232291856	3232289283	36919	808	TCP	1.000000e+00	0.000	1	58.0	attacker	portScan	0.000000

1000000 rows × 12 columns

Figure 1: Source Dataset: starting ip, ending ip, port number, traffic occurrence time, traffic end time, number of packets and bytes transmitted (Please note that for clarity, the figure is consistent with that in the previous report).

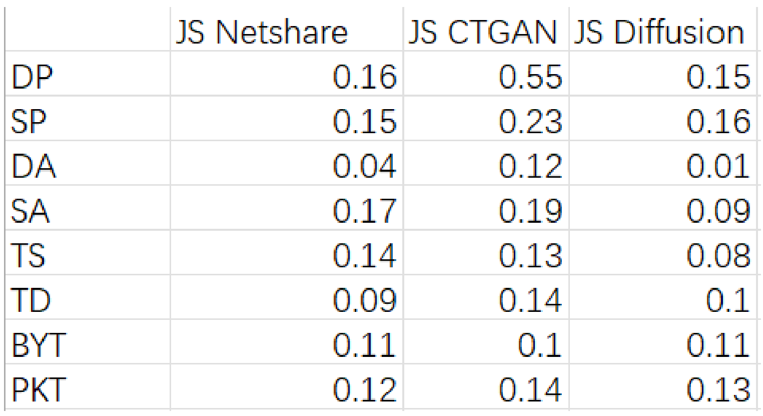
- Nethsare[21]: The nethsare framework uses a time series GAN for the generation of network traffic data, while it focuses on the temporal characteristics of the network traffic data, between the inputs to the GAN framework, it divides the network traffic data into n chunks with temporal relationships and inputs them sequentially into the framework of the time series GAN.

4.3 Evaluation

We use two evaluation modes for evaluation, the first being the use of evaluation metrics. Specifically, we use JS evaluation metrics for evaluation. The JS metric measures the distribution between the data we generate and the real data. The second one is that we use downstream tasks for evaluation.

$$JS(P \parallel Q) = \frac{1}{2}(KL(P \parallel M) + KL(Q \parallel M))$$

Specifically, we construct a predictive classification task. First, we trained a network using raw web traffic data and recorded the prediction error MSE values. Then, we fed the generated data into this network for prediction and recorded the prediction error MSE values. By comparing the difference in task performance between the generated data and the raw data, we can assess the quality of the generated data. This aims to ensure that our generated data has similar accuracy to the real dataset in different categorization and follow-up tasks. Suppose there is no significant difference in task performance between the generated and original data in this simple classification task. In that case, it indicates that the quality of the generated data is good. In summary, this downstream task can be used as a simple and effective method to verify the quality of generated data.



	JS Netshare	JS CTGAN	JS Diffusion
DP	0.16	0.55	0.15
SP	0.15	0.23	0.16
DA	0.04	0.12	0.01
SA	0.17	0.19	0.09
TS	0.14	0.13	0.08
TD	0.09	0.14	0.1
BYT	0.11	0.1	0.11
PKT	0.12	0.14	0.13

Figure 2: JSD Compare

4.4 Future work I: Diffusion Modeling for Table Generation

Several studies have focused on using diffusion models for tabular data generation, mainly in medicine, on generating patients' medical records and then performing subsequent recognition tasks or assisting physicians' judgment. In this category, most of them first use some priori-based embedding methods and then divide the generation into two categories: continuous data generation and discrete data generation. At the same time, some functions are added to help training, including but not limited to adding a comparison learning module. Some research like this is less available in other areas, such as computer security, and we would like to try to use the diffusion model to separate the generation of continuous feature dimensions in the network traffic data and discrete feature dimensions in the network traffic data. We suspect this will give better results, but the training time will also extend. Therefore, how to balance the two will be the focus of improvement. In addition, we found that the fine-graininess of the generated data needs to be more precise for either GAN or diffusion model generation, and we would like to generate more detailed data that is not noisy.

4.5 Future work II: Deployment (operator optimization)

Diffusion models are famous for their higher accuracy, greater arithmetic consumption, and smoother training convergence process compared to other generative models. Researchers have invested significant effort in exploring methods to accelerate the training of diffusion models. Despite these endeavors, advancements in the mathematical aspects of diffusion modeling have been relatively slow compared to the rapid progress in deployment optimization in the industry. As a result, developing a comprehensive deployment scheme has become crucial to exploit the potential of diffusion models fully. An efficient deployment scheme aims to define the matrices involved in the diffusion modeling process and optimize the matrix computations for faster execution on modern hardware, particularly Graphics Processing Units (GPUs). GPUs have proven instrumental in accelerating deep learning tasks due to their parallel processing capabilities. For diffusion models, deploying the computations to GPUs can significantly speed up the training and inference process, allowing for quicker generation of high-quality samples.

References

[1] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul. Generative deep learning to detect cyberattacks for the iot-23 dataset. *IEEE Access*, 10:6430–6441, 2022.

[2] Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. On the complexity of traffic traces and implications. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1):1–29, 2020.

[3] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.

[4] Lelio Campanile, Marco Gribaudo, Mauro Iacono, Fiammetta Marulli, and Michele Mastroianni. Computer network simulation with ns-3: A systematic literature review. *Electronics*, 9(2):272, 2020.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[6] Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaarfar, and Haojin Zhu. Differentially private data generative models. *arXiv preprint arXiv:1812.02274*, 2018.

[7] Adriel Cheng. Pac-gan: Packet generation of network traffic using generative adversarial networks. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0728–0734, 2019.

[8] Qiumei Cheng, Shiying Zhou, Yi Shen, Dezhang Kong, and Chunming Wu. Packet-level adversarial network traffic crafting using sequence generative adversarial networks, 2021.

[9] Ramin Hasibi, Matin Shokri, and Mehdi Dehghan. Augmentation scheme for dealing with imbalanced network traffic classification using deep learning. *arXiv preprint arXiv:1901.00204*, 2019.

[10] Zhicheng Hu, Jianqi Shi, YanHong Huang, Jiawen Xiong, and Xiangxing Bu. Ganfuzz: a gan-based industrial network protocol fuzzing framework. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 138–145, 2018.

[11] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187, 2002.

[12] Zinan Lin, Hao Liang, Giulia Fanti, and Vyas Sekar. Raregan: Generating samples for rare classes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7506–7515, 2022.

[13] Fabien Meslet-Millet, Sandrine Mouysset, and Emmanuel Chaput. Necstgen: An approach for realistic network traffic generation using deep learning. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 3108–3113, 2022.

[14] Hung Nguyen-An, Thomas Silverston, Taku Yamazaki, and Takumi Miyoshi. Generating iot traffic: A case study on anomaly detection. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, pages 1–6, 2020.

[15] Mustafizur R. Shahid, Gregory Blanc, Houda Jmila, Zonghua Zhang, and Hervé Debar. Generative deep learning for internet of things network traffic generation. In *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 70–79, 2020.

[16] Leonel Tedesco, Aline Mello, Diego Garibotti, Ney Calazans, and Fernando Moraes. Traffic generation and performance evaluation for mesh-based nocs. In *Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 184–189, 2005.

[17] Ly Vu, Cong Thanh Bui, and Quang Uy Nguyen. A deep learning based method for handling imbalanced problem in network traffic classification. In *Proceedings of the 8th international symposium on information and communication technology*, pages 333–339, 2017.

[18] Pan Wang, Shuhang Li, Feng Ye, Zixuan Wang, and Moxuan Zhang. Packetcgan: Exploratory study of class imbalance for encrypted traffic classification using cgan. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–7, 2020.

[19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Shengzhe Xu, Manish Marwah, and Naren Ramakrishnan. STAN: synthetic network traffic generation using autoregressive neural models. *CoRR*, abs/2009.12740, 2020.

[21] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 458–472, 2022.