

Testbed-Driven Class-Aware Synthetic Data Augmentation for 5G DDoS Detection

Nasim Nezhadsistani, Cédric Ivo Lüchinger, Lorenzo Ladner, Cyrill Georg Meier, Weijie Niu, Burkhard Stiller,
Communication Systems Group, Department of Informatics, University of Zürich, Switzerland
Email: {nezhadsistani, niu, stiller}@ifi.uzh.ch, {cedricivo.luechinger, lorenzo.ladner, cyrillgeorg.meier}@uzh.ch

Abstract—Softwarization of fifth-generation (5G) core networks enables flexible, programmable service deployment but also makes critical control and user plane functions vulnerable to DDoS attacks. The strong potential of machine learning-based intrusion detection is constrained in 5G environments by the scarcity of realistic datasets and the severe class imbalance in existing benchmarks. This paper presents a testbed-driven framework for generating and augmenting 5G DDoS detection data. The framework builds on a lightweight, free5GC- and UERANSIM-based environment with explicit control-plane/user-plane separation, a Mininet-based dataplane fabric, and traffic instrumentation across the N2, N3, N4, and N6 interfaces. The CEDI-NIDD dataset is combined with the public 5G-NIDD benchmark to evaluate CTGAN and TabDDPM under balanced and class-aware augmentation strategies. Results show that class-aware generation is critical: per-class augmentation improves 5G-NIDD accuracy from 0.721 to 0.984 with TabDDPM and up to 0.9999 with CTGAN and LightGBM, whereas a single balanced generator provides no measurable improvement over the non-augmented baseline. These results show that balancing sample counts alone is insufficient for effective synthetic augmentation in 5G intrusion detection; preserving class-specific traffic distributions is necessary.

Index Terms—5G security, DDoS detection, intrusion detection, synthetic data augmentation

I. INTRODUCTION

5G network technology is progressively utilized for mission-critical as well as extensive applications such as enhanced mobile broadband, industrial automation, intelligent transportation systems, and massive IoT. This advancement is made possible due to the software-defined core network that is characterized by virtualization of network functions, their programming, and interconnectivity via standardized service-based interfaces. Although this approach brings benefits regarding flexibility, scalability, and automation, it makes the attack surface of mobile networks larger as well. Namely, the splitting of control plane and user plane into separate functions creates various opportunities for an adversary to overload signaling, forwarding, or session management processes.

DDoS attacks still represent one of the biggest problems with network availability. In the case of 5G networks, DDoS attacks do not only concern classical volumetric flooding of the user plane. Such attacks can also target the network control plane and other elements related to the network core functionality, including the Access and Mobility Management Function (AMF), Session Management Function (SMF), and User Plane Function (UPF). Therefore, low-rate DDoS attacks on the

control plane or network core interface may cause problems with session establishment and user access.

Machine learning (ML) and deep learning (DL) techniques have been widely investigated for network intrusion detection, as they can model complex traffic patterns and identify malicious behavior from high-dimensional features. However, their practical use in 5G security is strongly limited by the availability and quality of training data. Many commonly used intrusion detection datasets, such as NSL-KDD, UNSW-NB15, and CIC-IDS2018, were not generated over 5G infrastructures and therefore do not capture 5G-specific protocol behavior, core-network interfaces, or tunneling characteristics [1]–[3]. The 5G-NIDD dataset represents an important step toward realistic 5G intrusion detection research, since it was generated over a functional 5G test network [4]. Nevertheless, it remains highly imbalanced, which can bias classifiers toward majority classes and lead to poor detection of rare attacks.

Generating synthetic data appears to be an interesting research direction that could help overcome this disadvantage. Rather than applying only random oversampling or interpolation techniques, modern tabular data generation models can capture dependencies between features and produce new samples from minority classes. There have been attempts to use conditional GANs, including CTGAN and CTAB-GAN, for synthesizing mixed-type tabular data with an imbalanced or long-tail distribution [5], [6]. The recent CTAB-GAN+ further improved tabular data generation by focusing on the utility and statistical similarities and adding privacy awareness to the generation process [7]. Tabular data diffusion models, like TabDDPM, have demonstrated outstanding results in heterogeneous data generation [8]. However, application of such approaches to 5G intrusion detection is still understudied, specifically with respect to the effect of the augmentation strategy on detection performance.

This study bridges the aforementioned gap through the introduction of a testbed-based framework for synthetic data augmentation in 5G DDoS attacks. This framework can be seen as comprising two major modules. Firstly, a repeatable 5G testbed, which uses free5GC, UERANSIM, and Mininet for the creation of labeled benign and malicious traffic across core network interfaces is applied. Secondly, a synthetic data augmentation and classification pipeline compares the effectiveness of CTGAN and TabDDPM in two data augmentation schemes: a balanced strategy based on a single global generator, and a per-class strategy where separate generative models

are trained for individual traffic classes.

The main contributions of this paper are summarized as follows:

- CEDI-NIDD is introduced as a 5G intrusion detection dataset that complements existing 5G-NIDD data by focusing on core-network DDoS scenarios.
- Comparison of CTGAN and TabDDPM approaches for tabular synthetic data augmentation in balanced and per-class augmentation settings.
- The results demonstrate that per-class augmentation substantially improves detection performance on 5G-NIDD, while balanced augmentation with a single global generator does not improve over the non-augmented baseline.

The implementation code is publicly available at: project's GitHub repository. The remainder of this paper is organized as follows. Section II presents the system design and Section III the methodology. Section IV covers the experimental setup and results. Finally, Section IV-C concludes the paper.

II. SYSTEM DESIGN

DDoS detection is evaluated over a controlled, lightweight end-to-end 5G Standalone (SA) testbed with a 5G core and its explicit control-/user-plane separation [9]. The framework is organized in two decoupled stages that can be operated independently. Stage I emulates the 5G network and generates labeled benign and attack traffic. While Stage II consumes the captured traffic as a ML/DL detection pipeline.

Stage I: The testbed extends the lightweight mini-5g environment. Mini-5g couples free5GC and UERANSIM but places the 5G-core behind a NAT and omits the N6 interface, departing from the 3GPP reference architecture. The topology is redesigned to track the 3GPP SBA more closely while remaining deployable on a single commodity laptop. The deployment spans two virtual machines: a control VM hosting all 5G core control-plane functions, and a data VM running a Mininet fabric that hosts the User Plane Function (UPF), the RAN (UERANSIM UE + gNB), and the Data Network (DN). The UPF is deliberately placed on the data VM, to enforce strict Control/User-Plane Separation (CUPS) and allow the two planes to be captured and analyzed independently. Detailed information of the topology can be found at the Table I.

TABLE I: Legend of Fig. 1

Abbreviation	Explanation
h1	UE and gNB
h2	UPF, (forwarding, internet gateway)
h3	DN, (web-, streaming-, game-server)
h4	Adversary, for attack experiments
r0	Capturing point N2/N3, in GTP-U tunnel
r1	Router between UPF and DN, capturing point N6
Data VM	Mininet host running six different hosts
Ctrl VM	5G core host
GTP-U	Encapsulation tunnel for user data

Four 3GPP interfaces are instrumented in this project: N2 (gNB↔AMF, NGAP over SCTP), N3 (gNB↔UPF, GTP-U user-plane tunnel), N4 (SMF↔UPF, PFCP), and N6 (UPF↔DN). The Traffic is captured at three independent

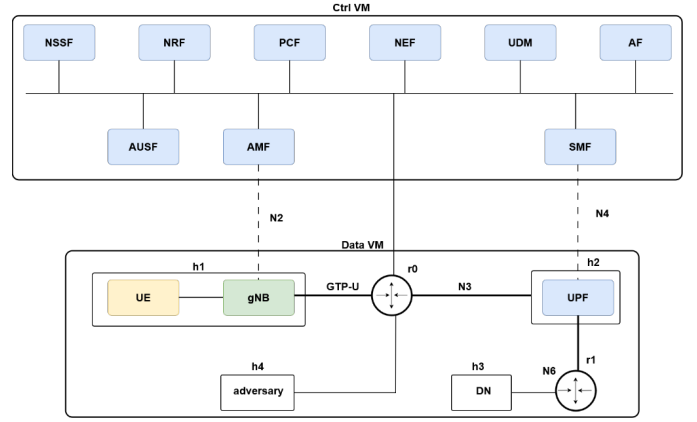


Fig. 1: Service-based view of the 5G framework

measurement points. First at router r0, on the gNB–UPF path inside the GTP-U tunnel (N2/N3). Second at router r1, on the UPF–DN path (N6, de-capsulated). And third the control VM (N4 and core-internal signaling). Capturing N3 at r0 and N6 at r1 simultaneously enables a direct comparison of GTP-encapsulated user data against its de-capsulated counterpart. The described service-based view of the 5G framework can be found in Figure 1. For the threat model, an adversary is considered capable of injecting traffic at distinct points of the 5G core and targeting both the control and user planes. This extends prior user-plane DDoS analyses on 5G network slices [10] to control-plane core interfaces (N2, N4) alongside user-plane ones (N3, N6). Four single-vector flooding attacks are modeled, each targeting one interface (summarized in Table III). First N2 (control plane) attack, a malicious RAN endpoint (the gNB host) floods the AMF with SCTP/NGAP packets, stressing registration/mobility signaling. Second attack is on N3 (user plane). The gNB host floods the UPF through the GTP-U tunnel with GTP-encapsulated UDP, TCP, or ICMP packets. For N4 flooding (control plane), a compromised SMF floods the UPF with PFCP session-management packets. On the N6 interface (user plane), an external/on-path adversary floods the DN with source-IP-randomized TCP traffic toward the web service (port 80). The adversary is assumed to have on-path or endpoint access sufficient to emit these flows but no ability to break 5G-AKA authentication or decrypt user-plane traffic. Physical-layer attacks (e.g., jamming) and air-interface-only vulnerabilities are out of scope. The raw packet captures produced in Stage I are truncated, anonymized, and exported into Stage II.

Stage II: Stage II is the machine-learning detection pipeline that consumes the captured traffic. It is organized as a five-step, work-package-based architecture (Figure 2): the two datasets are ingested (WP1), cleaning and 80/20 splitting (WP2), construction of the Raw, Dirty, Balanced, and augmented sub-datasets (WP3), synthetic generation with CTGAN and TabDDPM under balanced and per-class strategies (WP4), and finally used to train and evaluate gradient-boosted classifiers (WP5). The public 5G-NIDD benchmark [4] drives

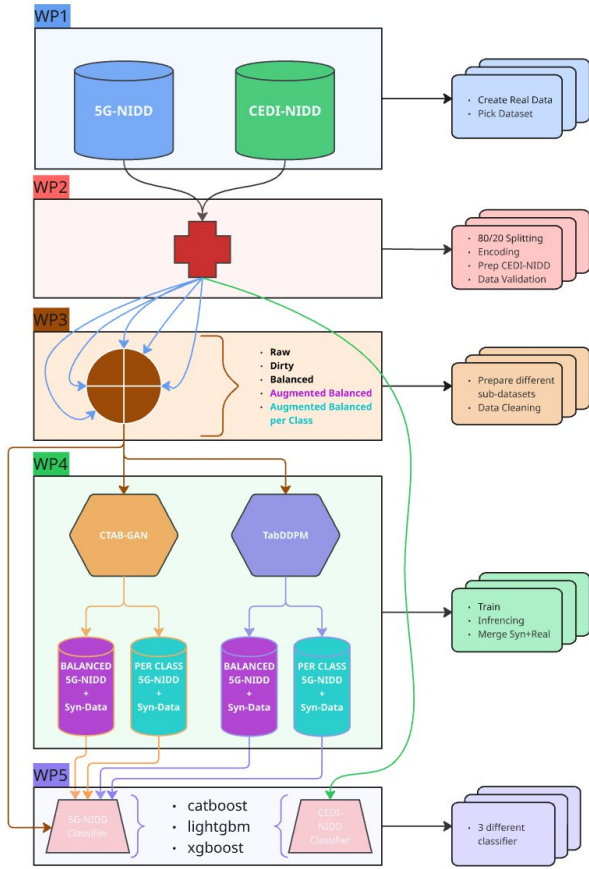


Fig. 2: Stage II: augmentation-and-classification pipeline.

the augmentation study, while the CEDI-NIDD captures from Stage I follow the same preprocessing and classification path but bypass augmentation and provide an independent cross-dataset validation. The two datasets are never mixed, as they do not share a feature space.

III. METHODOLOGY

Stage I: Simulation environment. The framework is built exclusively from open-source components. Found in II, free5GC v3.2.1 [11] provides the SA core and was selected over Open5GS for its native SA support and tested interoperability with UERANSIM [12] (commit 3a96298). Mininet [13] emulates the data-VM fabric, with each host on a separate VLAN. The VMs are provisioned reproducibly via Vagrant over VirtualBox.

The environment was deployed on a commodity laptop (Intel Core i5, 16 GB RAM, Windows 11). The testbed has four benign (mobile gaming, web-browsing, ping, and streaming) and four attack (N2, N3(TCP, UDP, ICMP), N4, N6) scenarios pre-implemented. For benign traffic, mobile gaming traffic consists a game server and client. They mimicking turn-based mobile gaming via TCP sockets using port 5555. On the client, a jitter is implemented using Gaussian randomization and a random spike for realism in delay. The base of the delay is 30 ms with a standard deviation of 10 ms. A spike has a

TABLE II: 5G Testbed Software Stack (Open-Source)

Component	Version	Role
free5GC	3.2.1	5G SA core
UERANSIM	3.2.6-48-g3a96298	RAN simulation (UE + gNB)
Mininet	≥2.3.1b4	Data-VM network emulation
Vagrant	≥2.4.6	VM provisioning
VirtualBox	≥7.0.22	Hypervisor
VcXsrv	≥21.1.16.1	X11 server for Mininet host CLI
Node.js	≥20.19.6	free5GC web-console runtime
ffmpeg	≥4.2.7	Streaming
puppeteer	≥10.8.2	Browser automation (web traffic)
nginx	≥1.18.0	DN web server
tcpreplay	≥4.3.2	Replay-based flooding (N2/N3/N4)
hping3	≥3.0.0-alpha-2	Packet crafting / flooding (N6)
tcpdump	≥4.9.3	Packet capture

TABLE III: DDoS Attack Scenarios per 5G Interface

Iface	Plane	Target	Vector	Tool
N2	Control	AMF	SCTP/NGAP flood	tcpreplay
N3	User	UPF	GTP-U {UDP,TCP,ICMP} flood	tcpreplay
N4	Control	UPF	PCFP flood (from SMF)	tcpreplay
N6	User	DN	TCP flood, randomized src IP	hping3

duration of 120 ms with a probability of 5%. For realistic web-browsing, Puppeteer was used for randomized tab opening (1-5 tabs) and closing with simulated reading time (between 1 and 10 minutes). The nginx web server in the DN is getting visited. For the simulation of streaming, the client receives the streams, and the server sends the streams concurrently, video and audio. The threads send via UDP sockets MPEG-4 Part 14 (MP4) video streams to port 5000 of the client and MP4 audio to port 5001 of the client. The client sink-holes the received streams. Since the purpose of creating traffic has already been met. FFmpeg was used to stream the video and audio. Using Real-time Transport Protocol (RTP). To mimic Web Real-Time Communication (WebRTC), the primary format, libopus, was used for encoding audio, with a bitrate of 128kbps. The resolution was set to 720p. Deployed with H.264 codec. And lastly a standard ping was done. All benign traffic simulations were done for exactly 4 hours. For attack traffic, tcpreplay was used to replay previously captured packets. Except for N6 flooding where hping3's flooding option was used. To capture the complete traffic, the rate of N4- and all N3-floodings needed to be reduced. In the case of N4 to 500 Mbps. And in the case of all N3 floodings to 45 Mbps. In the case of the N2 flooding, h1 sends SCTP packets to the AMF of the ctrl-vm. For all three types of N3 floodings resend GTP encapsulated packets (GTP/UDP, GTP/TCP, GTP/ICMP) from h1 to h2 (UPF) through the dedicated GTP-Tunnel. To conduct the N4 flooding, PCFP packets were sent from the SMF of the ctrl-vm to the UPF in the data-vm. N6 flooding targeted port 80 (HTTP) on h3 with an NGINX web server deployed. A complete overview can be found in the Table III. Each individual attack was conducted with a duration of 3 minutes (short), 6 minutes (medium), and 10 minutes (long). Dataset construction and feature extraction. Captures are anonymized by truncating packets to 96 bytes, and pseudonymizing MAC addresses with tcprewrite (seed 42). IP remapping was unnecessary as all addresses are syn-

TABLE IV: CEDI-NIDD Dataset: Main Parameters

Parameter	Value
File format	PCAP
Number of files	26
Benign	8
Attack	18
Capture points	r0 (in GTP-U tunnel), r1, ctrl-vm
Traffic categories	Benign, flooding
Total size	51.2 GB

TABLE V: CEDI-NIDD: Truncated and Anonymized Captures

File	Scenario	Packets	Time (s)	Size (MB)
<i>Benign traffic</i>				
gaming_r0.pcap	GTP/TCP	1,793,742	14,400	200
gaming_r1.pcap	TCP	1,790,424	14,400	165
ping_r0.pcap	GTP/ICMP	42,916	14,400	4.806
ping_r1.pcap	ICMP	38,236	14,400	4.282
streaming_r0.pcap	GTP/UDP	14,583,521	14,400	1,407
streaming_r1.pcap	UDP	10,547,352	14,400	1,099
webbrowsing_r0.pcap	GTP/HTTP	4,620	14,400	0.517
webbrowsing_r1.pcap	HTTP	790	14,400	0.071
<i>Attack traffic</i>				
n2_flooding.pcap	SCTP	40,795,292	600	4,079
n3_udp_flooding.pcap	GTP/UDP	8,763,825	600	981
n3_tcp_flooding.pcap	GTP/TCP	8,293,347	180	928
n3_icmp_flooding.pcap	GTP/ICMP	23,576,899	360	2,640
n4_flooding.pcap	PFCP	1,761,924	600	151
n6_flooding.pcap	TCP	4,983,055	180	358

r0: captured inside the GTP-U tunnel; r1: after decapsulation.

thetic/emulated. The resulting CEDI-NIDD dataset comprises 26 raw PCAP files (8 benign, 18 attack; 51.2 GB total) captured at r0, r1, and the control VM. The main parameters can be found in Table IV.

Owing to resource limits, a processed subset was retained, and a few attack scenarios fell back to shorter durations (N3-TCP and N6 to 3 min; N3-ICMP to 6 min). The streaming files are the biggest in size and number of packets. There was always more traffic captured at r0 than in r1. Notable observations in the captured traffic are that both gaming files captured retransmissions caused by the jitter and spike implemented. IPv4 fragmentation are found in the streaming traffic. The reason is that the Maximum Transmission Unit (MTU) limit is hit. The N4 flooding attack succeeded every time after only two PFCP packets sent. Causing the UE to disconnect completely from the network and SCTP abort packets being captured. A closer look at the N11 interface is needed. The N11 interface sits between the AMF and SMF in the ctrl-vm. However this is left aside for future research. The overview of the captures can be found in Table V.

Stage II: Stage II follows a controlled-experimentation methodology. Every preprocessing, augmentation, and classification step is held fixed across conditions, so that performance differences can be attributed to the data rather than to configuration changes.

Preprocessing. After an 80/20 stratified split, a feature-cleaning step removes columns that leak the label or are redundant (network-layer identifiers, TCP timing and window fields, VLAN identifiers, and rate/gap derivatives). Heavy-

tailed continuous features are log-transformed, and all numeric features are standardized, which also stabilizes the generative models. Three non-augmented references are derived from the cleaned training pool: a *Raw* variant preserving the original imbalanced distribution, a *Dirty* variant retaining the uncleaned encoded columns, and a *Balanced* variant in which each of the eight attack types is resampled to 10,000 records and the benign class matched to the 80,000 malicious total, giving a 160,000-row pool with a 50/50 benign/malicious split.

Generative models. Two generators from different paradigms are compared, so that an observed effect can be attributed to augmentation in general rather than to a single model: CTGAN [5], a conditional GAN for mixed-type tabular data, and TabDDPM [8], a denoising diffusion model for tabular inputs. Binary one-hot columns are declared discrete so that the generators respect their range.

Augmentation strategies. Two strategies are evaluated for each model. The *balanced* strategy trains a single global generator on the balanced pool, learning a joint distribution over all classes. The *per-class* strategy trains a separate generator for each traffic class and concatenates the per-class outputs; this is motivated by the hypothesis that class-isolated generation better preserves the intra-class structure of minority attacks. In both strategies the synthetic samples are merged with the real data at a 50/50 ratio, yielding augmented training sets of 320,000 rows. Together with the three references, this produces seven 5G-NIDD training configurations.

Classification. Each configuration is classified with three gradient-boosted decision-tree (GBDT) models: XGBoost [14], LightGBM [15], and CatBoost [16] under a flat multi-class formulation in which each flow is assigned one attack type or the benign class. Feature sets are aligned to their common intersection, giving a shared 40-feature input vector, and a 10% internal validation split enables early stopping. Training all three classifiers on every configuration controls for model-specific bias.

IV. EXPERIMENTAL EVALUATION

A. Testbed Setup and Experiment Description

Datasets. The augmentation study uses the public 5G-NIDD dataset [4], which contains 1,215,885 labeled flow records spanning nine classes: one benign class and eight attack types (UDPFlood, HTTPFlood, SlowrateDoS, TCPConnectScan, SYNScan, UDPScan, SYN Flood, and ICMP Flood). It is consumed in its officially provided encoded form. The dataset is severely imbalanced: the benign and UDPFlood classes together account for roughly 77% of all records, whereas ICMP Flood represents under 0.1%, an imbalance exceeding 400:1 in the held-out test set. CEDI-NIDD (Tables IV and V) is used as an independent dataset to validate the classification pipeline on traffic captured on the proposed testbed.

Environment and configuration. The Stage II pipeline runs on a workstation with Ubuntu 24.04 (Python 3.12.3) using GPU acceleration (NVIDIA CUDA) for both generator and classifier training. CTGAN is trained for 500 epochs with a

batch size of 500 and a PacGAN factor of 10; TabDDPM is trained for 500 iterations with a batch size of 500 and a learning rate of 0.002. Both generators produce 160,000 synthetic records per augmentation run, which are merged 50/50 with the real data. All random seeds are fixed for reproducibility.

Evaluation protocol. Augmentation utility is assessed under an evaluation on a held-out real test set protocol: each of the seven 5G-NIDD configurations is classified and evaluated on the corresponding held-out real test set, and CEDI-NIDD is trained and evaluated independently. Detection is reported with overall and balanced accuracy; macro-, weighted-, and micro-averaged precision, recall, and F1; the Matthews Correlation Coefficient (MCC) and Cohen’s κ ; and the benign false-positive rate (FPR). Synthetic-data fidelity is assessed separately with six metrics comparing real and synthetic samples: the average Kolmogorov-Smirnov (KS) statistic and Wasserstein distance (WD) over numeric features, the average Total Variation Distance (TVD) over binary features, the Frobenius norm of the difference between real and synthetic correlation matrices, the PCA coverage ratio, and the Maximum Mean Discrepancy (MMD).

B. Synthetic Data Quality

Before measuring downstream utility, the fidelity of the synthetic data is evaluated for all four augmentation configurations (Table VI). The two generators show a consistent trade-off. TabDDPM reproduces individual marginal distributions more closely, achieving a lower average KS statistic in both the balanced (0.266 vs. 0.422) and per-class (0.328 vs. 0.496) settings. CTGAN, however, is markedly stronger on the structural and joint-distribution metrics: it attains lower Wasserstein distance, lower TVD on binary indicators, a closer correlation structure, and, most decisively, MMD values one to two orders of magnitude lower than TabDDPM (0.009 vs. 0.220 balanced; 0.007 vs. 0.452 per-class). Per-class training improves PCA coverage for both models, narrowing TabDDPM’s manifold gap (0.683 to 0.880) and slightly improving CTGAN (0.817 to 0.888). A weighted comparison across the six metrics, giving higher weight to the joint-distribution metrics (MMD, KS) and the structural metrics (PCA coverage, Frobenius distance), confirms this picture: CTGAN wins on five of the six metrics in both strategies, with TabDDPM’s only advantage being the marginal KS statistic. TabDDPM’s samples are thus individually plausible per feature but occupy regions of the joint feature space that the real data does not, whereas CTGAN preserves the joint structure more faithfully, the property that matters more for training a detector. This manifold gap is visible in the PCA projections of Fig. 3.

C. Detection Performance

Table VII reports representative configurations from the 21 model configuration combinations on 5G-NIDD; the results stratify sharply by augmentation strategy.

Per-class augmentation drives the gains. Per-class augmentation raises overall accuracy far above both the non-

TABLE VI: Synthetic-data fidelity across augmentation configurations.

Config.	KS	WD	TVD	Frob.	PCA	MMD
CTGAN, Bal.	0.422	0.062	0.021	1.539	0.817	0.009
TabDDPM, Bal.	0.266	7.130	0.029	2.893	0.683	0.220
CTGAN, PC	0.496	0.078	0.007	1.654	0.888	0.007
TabDDPM, PC	0.328	2.268	0.031	2.575	0.880	0.452

Note: Lower values are better for KS, WD, TVD, Frob., and MMD; higher values are better for PCA. Bal.: balanced; PC: per-class (mean across classes); Frob.: Frobenius distance; PCA: PCA coverage.

augmented baselines and balanced augmentation: from 0.721 at the best baseline to 0.984 with TabDDPM (consistent across all three classifiers) and up to 0.9999 for CTGAN per-class with LightGBM, which reaches near-perfect balanced accuracy, macro-F1, and MCC with zero benign false positives. TabDDPM per-class is the more stable option, varying by less than 0.0001 across classifiers, but its lower balanced accuracy (0.890) reflects more uneven per-class behavior and a lower ceiling than CTGAN’s best configuration.

Balanced augmentation provides no gain. Balanced augmentation does not improve over the non-augmented baselines: all baseline and balanced-augmentation runs cluster around 0.721 accuracy and 0.913 balanced accuracy. The gap between high balanced accuracy and low overall accuracy is driven by poor benign recall (0.407–0.531). Because the benign class is 39.3% of the test set, misclassifying roughly half of it depresses overall accuracy, while most attack classes remain well detected.

Interpretation. The near-perfect CTGAN per-class result should be read as an upper bound rather than an expected operational accuracy. It is highly classifier-specific; only LightGBM reaches 0.9999, while XGBoost and CatBoost on the same data reach 0.976 and 0.940, and the per-class merge together with possible memorization by the generator could inflate scores. Quantifying this memorization risk is left to future work.

Cross-dataset validation. Applying the same three classifiers to CEDI-NIDD without augmentation yields near-perfect baseline accuracy (above 0.9998 for all three, with a LightGBM balanced accuracy of 0.9857), indicating that the CEDI-NIDD traffic classes are highly separable in the extracted feature space and confirming the soundness of the shared feature-engineering and classification methodology.

This paper introduces a testbed-driven framework for synthetic data augmentation in 5G DDoS attack detection. The proposed framework integrates the reproducible and lightweight free5GC-UERANSIM testbed with control/user-plane separation, Mininet-based data plane, and network traffic instrumentation at the N2, N3, N4, and N6 interfaces. On top of that, the CEDI-NIDD dataset was created based on the proposed framework to represent benign and DDoS traffic from multiple 5G core interfaces and used alongside the public 5G-NIDD benchmark to compare various tabular synthetic augmentation methods. Experimental evaluation shows that the success of the proposed augmentation approach strongly depends on

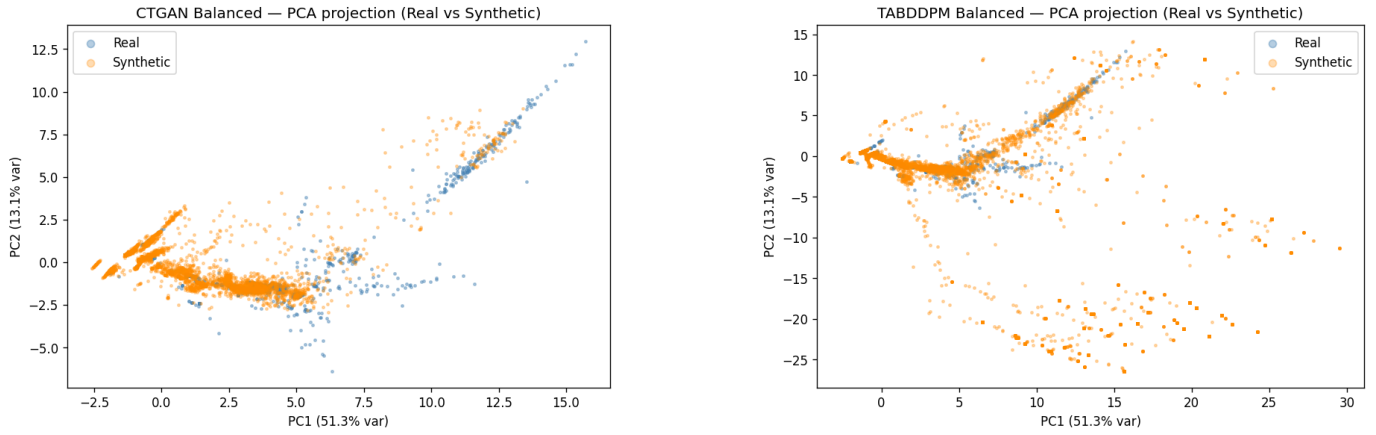


Fig. 3: PCA projection (PC1 vs. PC2) of real (blue) and synthetic (orange) samples for the balanced augmentation run.

TABLE VII: 5G-NIDD detection performance for representative configurations.

Configuration	Classifier	Acc.	Bal. Acc.	F1	MCC	κ	FPR
CTGAN per-class	LightGBM	0.9999	0.9999	0.9999	0.9999	0.9999	0.0000
TabDDPM per-class	LightGBM	0.9837	0.8900	0.8544	0.9766	0.9762	0.0000
CTGAN per-class	XGBoost	0.9758	0.9553	0.9630	0.9653	0.9646	0.0093
CTGAN per-class	CatBoost	0.9399	0.8885	0.8660	0.9169	0.9120	0.0000
Raw (best baseline)	XGBoost	0.7218	0.9131	0.8389	0.6266	0.6100	0.0606
CTGAN balanced (best balanced)	XGBoost	0.7211	0.9130	0.8379	0.6281	0.6091	0.0533
TabDDPM balanced (worst)	CatBoost	0.7173	0.9116	0.8296	0.6126	0.6038	0.0999

the generation strategy. Per-class generation strategy leads to significant improvements in terms of 5G-NIDD accuracy by improving its value from 0.721 to 0.984 with TabDDPM and up to 0.9999 with CTGAN and LightGBM. By contrast, balanced generation yields no measurable improvement over the original dataset. Indicating that balancing is only effective when combined with class-aware preservation of the traffic distribution. Future directions can be to quantify memorization in per-class generators, cross-dataset transfer, hybrid augmentation strategies and controlled synthetic data proportions.

ACKNOWLEDGMENT

This work was partially supported by the University of Zurich and the Horizon Europe NETWORK project (Grant No. 101139285), funded by SERI under Contract No. 23.00642.

REFERENCES

- [1] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.
- [2] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.
- [3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.
- [4] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, and M. Ylianttila, "5G-NIDD: A comprehensive network intrusion detection dataset generated over 5g wireless network," *arXiv preprint arXiv:2212.01298*, 2022.
- [5] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," *arXiv preprint arXiv:1907.00503*, 2019.
- [6] Z. Zhao, A. Kumar, H. V. der Scheer, R. Birke, and L. Y. Chen, "CTAB-GAN: Effective table data synthesizing," in *Proceedings of The 13th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 157. PMLR, 2021, pp. 97–112.
- [7] Z. Zhao, A. Kumar, R. Birke, H. V. der Scheer, and L. Y. Chen, "CTAB-GAN+: Enhancing tabular data synthesis," *Frontiers in Big Data*, vol. 6, p. 1296508, 2023.
- [8] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "TabDDPM: Modelling tabular data with diffusion models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 17564–17579.
- [9] 3GPP, "5G; System Architecture for the 5G System (5GS)," ETSI, Technical Specification TS 123 501, Jul. 2025.
- [10] M. S. Khan, B. Farzaneh, N. Shahriar, N. Saha, and R. Boutaba, "Slicesecure: Impact and detection of dos/ddos attacks on 5g network slices," in *2022 IEEE Future Networks World Forum (FNWF)*, 2022, pp. 639–642.
- [11] free5GC Project, "free5GC: Open-source 5g core network," <https://www.free5gc.org/>, 2025, accessed: 2026-06-21.
- [12] A. U. Contributors, "UERANSIM: Open source 5g ue and ran simulator," <https://github.com/aligungr/UERANSIM>, 2025, accessed: 2026-06-21.
- [13] Mininet Project Contributors, "Mininet: An instant virtual network on your laptop (or other pc)," <https://mininet.org/>, accessed: 2026-06-21.
- [14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.
- [15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 3146–3154.
- [16] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018, pp. 6638–6648.