

Resource-Aware Evaluation and Hybrid Learning with Synthetic Network Traffic for Intrusion Detection in Edge and IoT Environments

Neel Lapsiwala

Sardar Vallabhbhai National Institute of Technology Surat

Nishtha Srivastava

`nishtha.srivastava@nirmauni.ac.in`

Nirma University

Tanmoy Hazra

Sardar Vallabhbhai National Institute of Technology Surat

Nitesh Funde

Sardar Vallabhbhai National Institute of Technology Surat

Research Article

Keywords: Synthetic data generation, network intrusion detection, edge computing, Internet of Things, conditional variational autoencoder, conditional tabular GAN, resource-aware machine learning, robustness

Posted Date: October 7th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10637923/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Resource-Aware Evaluation and Hybrid Learning with Synthetic Network Traffic for Intrusion Detection in Edge and IoT Environments

Neel Lapsiwala^{1*}, Nishtha Srivastava², Tanmoy Hazra¹,
Nitesh Funde¹

^{1*}Department of Artificial Intelligence, Sardar Vallabhbhai National
Institute of Technology (SVNIT), Surat, Gujarat, India.

²Computer Science and Engineering Department, Institute of Technology,
Nirma University, Ahmedabad, 382481, Gujarat, India.

*Corresponding author(s). E-mail(s): i24ai002@aid.svnit.ac.in;
Contributing authors: nishtha.srivastava@nirmauni.ac.in;
tanmoyhazra@aid.svnit.ac.in; nitesh.funde@aid.svnit.ac.in;

Abstract

Deploying machine learning-based intrusion detection on Internet of Things (IoT) and edge devices is fundamentally constrained by limited compute, memory, and energy budgets, while the scarcity of labeled network traffic data further complicates model training. Unlike prior work that evaluates synthetic data primarily through statistical similarity metrics, this paper presents a deployment-aware evaluation framework that jointly assesses classification utility, inference efficiency, noise robustness, and energy consumption. We evaluate three data generation strategies on the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CIC-IDS-2017) benchmark under a Train-on-Synthetic, Test-on-Real (TSTR) protocol: a Random Forest trained on real data (serving as an upper-bound oracle whose real training data is *never* exposed to the synthetic generation pipeline at deployment), a Conditional Variational Autoencoder (CVAE), and a Conditional Tabular GAN (CTGAN). The CVAE jointly encodes class labels with feature vectors, eliminating the marginal label-sampling approximation of standard VAEs. We further validate generalizability on the University of New South Wales NB15 (UNSW-NB15) benchmark. Statistical significance of all pairwise performance differences is confirmed via McNemar’s test ($p < \mathbf{0.001}$), with all metrics reported as means $\pm$ 95% confidence intervals over five independent runs, and additionally under five-fold cross-validation on the synthetic training set.

The Random Forest oracle achieves 99.83% accuracy; the CVAE achieves 86.74% (outperforming the vanilla VAE at 83.11%), while CTGAN achieves 72.33%. We propose a hybrid prediction model grounded in multi-objective optimization, wherein criterion weights are selected via Bayesian optimization (Tree-structured Parzen Estimator, 100 trials) rather than manual assignment, yielding quality scores of 0.45, 0.43, and 0.12 for the Baseline, CVAE, and CTGAN, respectively. The oracle contributes only pre-trained inference scores at deployment; its real training data remains inaccessible to all synthetic learners throughout. The hybrid outperforms standard ensemble baselines (stacking, gradient boosting) on the composite resource-aware metric. Ablation and sensitivity analyses across 81 weight configurations confirm the stability of the weighting scheme. Practical deployment guidelines are provided for three hardware tiers: constrained IoT nodes, edge computing devices, and cloud/server infrastructure.

Keywords: Synthetic data generation, network intrusion detection, edge computing, Internet of Things, conditional variational autoencoder, conditional tabular GAN, resource-aware machine learning, robustness

1 Introduction

The rapid expansion of the Internet of Things (IoT) and edge computing has resulted in increasingly distributed infrastructures composed of sensors, cameras, embedded controllers, and other resource-constrained devices. By moving computation closer to data sources, edge computing can reduce communication latency and dependence on centralized cloud infrastructure; however, edge platforms typically operate under considerably tighter computational and memory constraints than conventional servers[1]. These constraints impose a fundamental challenge on machine-learning-based cybersecurity systems, which must achieve high detection performance while operating under a limited computational budget, and the rising number of connected devices makes the attack surface even larger.

Machine-learning-based network intrusion detection systems (NIDSs) are becoming increasingly popular techniques for detecting malicious activity on the network, such as scanning, denial-of-service attacks, exploitation attempts, and other anomalous traffic. The creation of useful learning-based NIDSs is, however, constrained by data availability, representativeness, and class imbalance of network traffic. Gathering and labelling a large amount of representative traffic from operational IoT/edge systems can be resource- and time-intensive; real-world network traffic often reveals sensitive device characteristics, system behaviors, and communication patterns. Such issues drive growing interest in utilizing synthetic data to increase and balance training-dataset size and to minimize the need for collecting actual network traffic[2].

Generative models, including Generative Adversarial Networks (GANs)[3], Conditional Tabular GANs (CTGANs)[4], and Variational Autoencoders (VAEs)[5], provide flexible mechanisms for learning complex data distributions and generating new samples. Such models are particularly relevant to network intrusion detection because network traffic datasets commonly contain heterogeneous features, highly imbalanced

attack classes, and complex relationships between traffic characteristics and intrusion categories. Conditional generative models can further incorporate class information into the generation process, potentially improving the semantic consistency of synthetic samples. More recently, diffusion-based[6] and language-model-based[7] tabular generators have expanded this space further, though their computational footprint may limit suitability for constrained deployment. Nevertheless, the practical value of generated data cannot be determined solely by examining whether synthetic samples resemble the real data distribution.

A central challenge in synthetic-data evaluation is therefore the distinction between *statistical fidelity* and *downstream utility*. Conventional fidelity measures can indicate whether generated data reproduce selected properties of real data, but do not necessarily establish whether a classifier trained on synthetic data will generalize to real traffic. Sample-level auditing frameworks[8] and tabular benchmarking studies[9] have separately shown that models competitive on aggregate statistical measures do not always transfer this advantage to downstream predictive tasks. This distinction is especially important for edge and IoT cybersecurity, where the final detection system must jointly satisfy high predictive performance, low inference latency, limited memory consumption, efficient CPU utilization, and robustness to input variability. Consequently, evaluation of synthetic data for intrusion detection should extend beyond statistical similarity to the operational characteristics of the resulting system.

Despite the growing literature on synthetic data generation and network intrusion detection, joint evaluation of synthetic-data utility and deployment constraints remains limited. Prior work has investigated generative approaches for structured and network data[10], including CTGAN-based intrusion detection augmentation[11], but a unified framework combining Train-on-Synthetic, Test-on-Real (TSTR) evaluation with hardware-aware resource measurements and robustness to input perturbations has received comparatively little attention. This gap matters because a generator that improves predictive performance may also increase computational requirements, while an efficient generator may yield insufficient downstream utility.

Our study bridges this gap with a deployment-aware framework assessing three related modeling approaches: a real-data-trained Random Forest as a high-level baseline; a CVAE; and a CTGAN. The synthetic-data-based models are evaluated using TSTR; the generated systems are then benchmarked using CIC-IDS-2017 and UNSW-NB15 in terms of prediction performance, inference latency, memory overhead, CPU utilization, and robustness to Gaussian noise perturbations over varying feature sets, class taxonomy, and traffic distributions[12, 13].

In addition to the model-level comparison, the resource-aware hybrid prediction framework is posed as a multi-criteria decision-making problem. Without manual tuning, Bayesian optimization, implemented using the TPE algorithm, selects appropriate values of combination weights in accordance with trade-offs between predictive accuracy, runtime, resilience, and CPU consumption[14]. Hence, the hybrid prediction scheme balances utility against practical constraints such as computational cost. This approach is compared against conventional stacking and gradient-boosting ensembles to assess whether multi-criteria optimization offers advantages over standard combination methods.

A strict separation is maintained between the real-data reference model and the synthetic-data pipeline: the Random Forest is trained solely on real data as an upper-bound reference and is never used to generate synthetic records, while the generators are trained independently. Accordingly, privacy-related claims are limited to reducing direct dependence on real records within the synthetic-data pipeline; no formal differential privacy guarantee is claimed.

The main contributions of this study are as follows:

- **Deployment-aware evaluation framework.** A unified framework integrating TSTR predictive utility with inference latency, memory consumption, CPU utilization, and robustness to input perturbations, providing a broader assessment of synthetic-data utility than statistical fidelity or accuracy alone.
- **Conditional generative modeling for intrusion detection.** A CVAE incorporating class information is compared against a vanilla VAE and CTGAN, improving TSTR accuracy from 83.11% to 86.74% on CIC-IDS-2017.
- **Resource-aware hybrid prediction.** Model combination is formulated as a multi-criteria optimization problem jointly considering predictive performance, computational efficiency, CPU utilization, and robustness, with a TPE-based search replacing manually selected weights.
- **Multi-dataset evaluation.** To study whether the developed framework can produce consistent results across different feature spaces, traffic distributions, attack types, and class imbalance levels, it is tested against the CIC-IDS-2017 and UNSW-NB15 datasets.
- **Statistical validation.** Repeated measurements, confidence intervals, cross-validation, and McNemar’s test are all used to determine whether the observed performance differences are statistically significant.
- **Deployment-oriented interpretation.** These results are then translated into resource-oriented guidance for different hardware layers, ranging from severely constrained IoT nodes to cloud and edge infrastructure. Physical hardware validation is bypassed by basing this guidance on computed hardware-resource characteristics.

The remainder of this paper is structured as follows: Section 2 reviews the background on generative models for synthetic network data, downstream usage of synthetic data, ensemble methods, and machine learning for resource-constrained networks and IoT device security. Section 3 introduces the problem formulation, dataset and the proposed framework. Section 4 describes the experimental evaluation, Section 5 presents the results, Section 6 discusses the findings, and Section 7 concludes the study and presents future work.

2 Related Work

2.1 Generative Models for Tabular Data

Generative Adversarial Networks (GANs)[3] offer a general scheme for learning data distributions via an adversarial interaction between a generator and a discriminator. Though GANs have proven effective for generating images, tabular data introduces

other concerns, including heterogeneity in continuous and discrete variables, mixed distributions of features, and severely imbalanced class proportions, leading to difficulty in learning to preserve marginal distributions and the dependencies between features. Conditional Tabular GAN (CTGAN)[4] aims to address these tabular challenges, adopting mode-specific normalization of continuous features and conditional sampling to represent discrete values and imbalanced classes, both of which are relevant to network intrusion detection, where heterogeneous features and unbalanced classes commonly occur. Some work has specifically aimed at generating network data; NetGAN[15] generates graph structures via random walks.

Flow-GAN[16] studied adversarial learning of network flow. Such work shows that generated network data can closely resemble real data in terms of statistics and structure, but whether a detector trained on the generated data will perform well in a resource-limited deployment remains uncertain. Recent diffusion-based[6] and language-model-based[7] tabular generators perform favorably on complex distributions, yet their computational and memory requirements restrict deployment on resource-constrained edge and IoT devices.

An alternative generative approach is based on modeling latent variables probabilistically (Variational Autoencoders, VAEs)[5], which learn a continuous latent representation and generate samples by decoding points in the latent space. The framework provides stable training and smooth interpolation of the latent space, and, with the help of conditioning on class labels, can be used for class-specific generation – of great importance in intrusion detection systems, since the dependence of traffic features on attack class needs to be accounted for during classification. This work uses conditional generation to improve synthetic-data utility for intrusion detection.

2.2 Synthetic Data Evaluation

The Train-on-Synthetic, Test-on-Real (TSTR) Protocol

The TSTR protocol[10] provides a task-oriented approach to measuring the practical utility of synthetic data: rather than focusing on statistical similarity, TSTR measures whether a model trained purely on synthetic data can generalize to real test data, since the main goal is not to faithfully recreate statistical traffic characteristics but to support meaningful downstream detection. Existing work tends to evaluate synthetic data in terms of classification performance metrics such as accuracy, precision, recall, and F1-score. However, such metrics alone cannot satisfy deployment requirements on resource-limited devices, since latency, memory usage, CPU load, and resilience to input variation all play critical roles in efficient and timely detection[17].

Alaa et al.[8] highlighted the importance of sample-level evaluation of synthetic data and its downstream performance, rather than relying solely on aggregated distributional statistics. Building on this insight, the TSTR protocol is extended here with four additional evaluation criteria: performance under Gaussian input perturbations, latency, memory usage, and CPU utilization, enabling synthetic-data methods to be evaluated jointly along accuracy, computational, and resilience dimensions.

2.3 Ensemble and Hybrid Methods

Ensemble learning combines predictions from multiple models to exploit complementary decision patterns: stacking combines base-learner outputs via a meta-model, while boosting constructs learners that progressively focus on prior errors. Although effective, conventional ensembles generally optimize predictive performance without explicitly incorporating component computational costs – a limitation particularly relevant when base models differ substantially in latency, memory, CPU utilization, or robustness, since a model improving accuracy may impose unacceptable costs in edge or IoT deployment. This study therefore treats model combination as a multi-criteria optimization problem rather than purely predictive aggregation.

Bayesian optimization offers an efficient framework for optimizing expensive, non-convex objectives over complex search spaces, with Tree-structured Parzen Estimators (TPEs) widely used for hyperparameter optimization[14]; recent work further highlights the importance of systematic optimization for complex ML pipelines[18]. Here, Bayesian optimization determines the relative weights assigned to predictive performance, computational efficiency, CPU utilization, and robustness, with the resulting hybrid model compared against stacking and gradient boosting to assess the value of explicitly incorporating deployment-related criteria into model combination.

2.4 Edge and IoT Machine Learning

Edge computing has emerged as an important paradigm for processing data closer to its source, with latency reduction, bandwidth limitations, and privacy identified as key motivating factors[1]. However, edge and IoT platforms operate under constrained computational, memory, and energy budgets, creating significant challenges for deploying ML-based security systems. Research in TinyML and memory-efficient machine learning has substantially reduced inference and training requirements[19, 20], yet comparatively little attention has addressed the interaction between resource constraints and synthetic training-data generation; the latter is often treated as an independent preprocessing task despite the generative model’s influence on downstream performance, computational requirements, and robustness. Existing research on intrusion detection datasets and IoT security further underscores the importance of jointly considering data characteristics, resource constraints, and deployment conditions[17], yet joint evaluation of synthetic-data utility, hardware-aware resource consumption, and robustness remains insufficiently explored in edge and IoT intrusion detection.

The present study addresses this gap at the intersection of synthetic-data generation, machine learning, and cybersecurity by proposing a unified framework combining TSTR-based predictive assessment with resource profiling and robustness analysis, together with a Bayesian-optimized hybrid prediction strategy integrating these deployment-relevant criteria into model combination. This extends prior work by treating synthetic-data generation not merely as a data-generation problem but as a component of an end-to-end pipeline for resource-constrained cybersecurity applications.

3 Methodology

This section presents the proposed resource-aware hybrid synthetic-data generation and evaluation framework for machine-learning-based network intrusion detection.

3.1 Problem Formulation

Let the preprocessed network intrusion dataset be defined as $\mathcal{D}_{\text{real}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ represents a d -dimensional network-traffic feature vector and $y_i \in \{1, \dots, C\}$ denotes its corresponding traffic or intrusion class.

The proposed framework contains three parallel modeling pathways. The first trains a Random Forest classifier directly on real labeled data, producing class-probability scores $\mathbf{p}_{\text{RF}}(\mathbf{x}) = f_{\text{RF}}(\mathbf{x})$; this model serves as a real-data reference (oracle) and is not used to generate synthetic training records. The second pathway learns a conditional generative distribution via a Conditional Variational Autoencoder, $G_{\text{CVAE}} : (\mathbf{z}, y) \rightarrow \tilde{\mathbf{x}}$, where $\mathbf{z}$ is a latent variable and y is the desired traffic class. The third pathway uses CTGAN to learn the distribution of mixed-type tabular network traffic, $G_{\text{CTGAN}} : \mathbf{z} \rightarrow (\tilde{\mathbf{x}}, \tilde{y})$.

For a generative model G_m with $m \in \{\text{CVAE}, \text{CTGAN}\}$, the synthetic dataset $\mathcal{D}_m^{\text{syn}} = G_m(\mathcal{D}_{\text{real}}^{\text{train}})$ is used to train a downstream classifier exclusively, which is then evaluated on real held-out traffic:

$$f_m^* = \text{Train}(\mathcal{D}_m^{\text{syn}}), \quad \hat{y} = f_m^*(\mathbf{x}), \quad (\mathbf{x}, y) \sim \mathcal{D}_{\text{real}}^{\text{test}}. \quad (1)$$

The framework therefore evaluates synthetic-data generation according to downstream predictive utility rather than statistical similarity alone.

3.2 Datasets

CIC-IDS-2017

The CIC-IDS-2017 dataset[12], developed by the Canadian Institute for Cybersecurity, is a widely used NIDS benchmark containing eight days of flow-level traffic with benign traffic and 14 attack categories (DoS Hulk, DDoS, PortScan, FTP-Patator, SSH-Patator, Bot, Web Attacks, Infiltration, among others). The original dataset contains 2,830,743 flow records and 79 features. After zero-variance feature removal, substituting infinity with NaN, and filtering out records with any missing values, 2,520,798 samples remain across 70 features and 15 traffic classes. 83% of samples are BENIGN, implying a large class imbalance. A stratified train-test split of 80:20 is performed with a seed of 42.

UNSW-NB15

UNSW-NB15[13], generated using the IXIA PerfectStorm platform at UNSW Canberra, serves as a second benchmark. It contains 257,673 records, 49 features, and 10 classes (Normal plus nine attack categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms). The same preprocessing and 80:20 stratified split are applied. Differences in scale, feature composition, class taxonomy, and class distribution relative to CIC-IDS-2017 provide a complementary evaluation setting.

These two datasets constitute the shared input to all three modeling pathways introduced below, and are used consistently throughout Stages 1–5 of the proposed framework.

3.3 Proposed Framework

As illustrated in Fig. 1, the framework follows a parallel generation architecture in which the preprocessed network traffic is provided simultaneously to three complementary modeling pathways: a real-data-trained Random Forest (RF) oracle, a Conditional Variational Autoencoder (CVAE), and a Conditional Tabular Generative Adversarial Network (CTGAN). The pipeline is organized into five stages: (i) preprocessing and feature engineering; (ii) parallel generation using the RF oracle, CVAE, and CTGAN; (iii) production of model outputs, i.e., RF probability scores and synthetic datasets; (iv) TSTR-based evaluation of synthetic-data utility; and (v) Bayesian optimization of the hybrid prediction mechanism that produces the final decision. The framework is

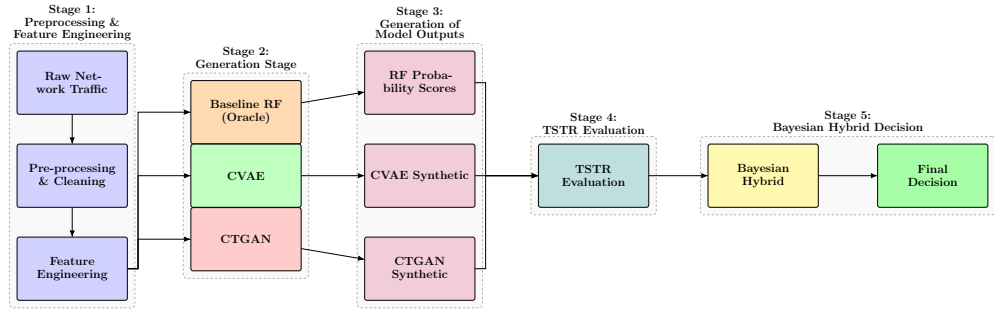


Fig. 1: Proposed framework.

designed to distinguish between the role of a real-data reference model and the role of synthetic-data generation. The RF model serves as a performance reference by learning directly from real labeled traffic, whereas the CVAE and CTGAN pathways generate synthetic network traffic for downstream evaluation. The synthetic-data pathways are therefore evaluated according to their ability to support detection of previously unseen real traffic rather than solely according to statistical similarity.

The complete workflow above is applied identically to both datasets described earlier and is detailed stage-by-stage in the following subsections.

3.3.1 Stage 1: Preprocessing and Feature Engineering

A common feature representation is supplied to all three pathways (RF, CVAE, CTGAN) to ensure a fair comparison. Zero-variance features are removed, infinite values are replaced with NaN, and incomplete records are discarded. Continuous features are standardized using training-set statistics:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j + \epsilon},$$

where x_j is the j -th continuous feature (raw value), x'_j is its standardized value, μ_j is the training-set mean of feature j , σ_j is the training-set standard deviation of feature j , and ϵ is a small positive constant added for numerical stability. Labels are encoded as integer identifiers (or one-hot vectors, where required by the conditional generative model). All preprocessing parameters (μ_j , σ_j) are estimated on the training partition only, to prevent leakage, yielding the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, where N is the number of samples and d is the feature dimensionality, for Stage 2.

3.3.2 Stage 2: Generation Stage

Baseline Random Forest Oracle

The RF pathway is trained directly on real labeled traffic, with predicted class-probability vector

$$\mathbf{p}_{\text{RF}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^T \mathbf{p}_t(\mathbf{x}),$$

where $\mathbf{x}$ is the input feature vector, $\mathbf{p}_{\text{RF}}(\mathbf{x})$ is the RF ensemble’s predicted class-probability vector, T is the number of trees in the forest, and $\mathbf{p}_t(\mathbf{x})$ is the class-probability vector produced by the t -th tree. It serves exclusively as a real-data reference: it does not generate synthetic records and does not expose real training samples to the CVAE or CTGAN pathways, contributing only frozen probability scores to the hybrid stage.

Conditional Variational Autoencoder

Given feature vector $\mathbf{x}$ and class label $\mathbf{y}$, the encoder estimates the posterior $q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$, where $\mathbf{z}$ is the latent representation and ϕ denotes the encoder’s trainable parameters. Latent sampling uses the reparameterization trick:

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}, \mathbf{y}) + \boldsymbol{\sigma}_\phi(\mathbf{x}, \mathbf{y}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $\boldsymbol{\mu}_\phi(\mathbf{x}, \mathbf{y})$ is the encoder’s predicted latent mean, $\boldsymbol{\sigma}_\phi(\mathbf{x}, \mathbf{y})$ is the encoder’s predicted latent standard deviation, $\odot$ denotes element-wise multiplication, and $\boldsymbol{\epsilon}$ is a noise vector sampled from the standard multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (zero mean, identity covariance). The decoder reconstructs the feature vector as $\hat{\mathbf{x}} = D_\theta(\mathbf{z}, \mathbf{y})$, where $\hat{\mathbf{x}}$ is the reconstructed feature vector, D_θ is the decoder function, and θ denotes the decoder’s trainable parameters. The training objective is

$$\mathcal{L}_{\text{CVAE}} = \mathcal{L}_{\text{rec}} + \beta D_{\text{KL}}[q_\phi(\mathbf{z} \mid \mathbf{x}, \mathbf{y}) \parallel p(\mathbf{z} \mid \mathbf{y})], \quad (2)$$

where $\mathcal{L}_{\text{CVAE}}$ is the total CVAE loss, $\mathcal{L}_{\text{rec}}$ is the reconstruction loss between $\mathbf{x}$ and $\hat{\mathbf{x}}$, β is a weighting coefficient controlling the strength of latent-space regularization, $D_{\text{KL}}[\cdot \parallel \cdot]$ is the Kullback–Leibler divergence, and $p(\mathbf{z} \mid \mathbf{y})$ is the class-conditioned prior over the latent space. Class-conditioning enables generation targeted at a specified intrusion category, allowing explicit control of the synthetic class distribution for imbalanced datasets.

Conditional Tabular GAN

CTGAN[4] trains a generator G and discriminator D under the adversarial objective

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] ,$$

where G is the generator network, D is the discriminator network, $\mathbf{x} \sim p_{\text{real}}$ denotes real samples drawn from the real data distribution p_{real} , $D(\mathbf{x})$ is the discriminator’s estimated probability that $\mathbf{x}$ is real, $\mathbf{z} \sim p(\mathbf{z})$ denotes a latent noise vector drawn from a prior distribution $p(\mathbf{z})$, and $G(\mathbf{z})$ is the generator’s synthetic sample produced from $\mathbf{z}$. CTGAN incorporates conditional sampling and mode-specific normalization for heterogeneous, imbalanced traffic features. Training yields the synthetic dataset

$$\mathcal{D}_{\text{CTGAN}}^{\text{syn}} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i=1}^M ,$$

where $\tilde{\mathbf{x}}_i$ is the i -th synthetic feature vector, $\tilde{y}_i$ is its corresponding synthetic class label, and M is the total number of generated synthetic samples.

The three pathways operate in parallel with distinct roles: RF produces reference probability scores, while CVAE and CTGAN produce synthetic datasets for downstream evaluation, preserving separation between the real-data reference and the synthetic pipeline.

3.3.3 Stage 3: Generation of Model Outputs

The generation stage outputs are collected as

$$\mathcal{O} = \{\mathbf{p}_{\text{RF}}, \mathcal{D}_{\text{CVAE}}^{\text{syn}}, \mathcal{D}_{\text{CTGAN}}^{\text{syn}}\} ,$$

where $\mathcal{O}$ is the combined output set, $\mathbf{p}_{\text{RF}}$ is the RF oracle’s probability scores, $\mathcal{D}_{\text{CVAE}}^{\text{syn}}$ is the CVAE-generated synthetic dataset, and $\mathcal{D}_{\text{CTGAN}}^{\text{syn}}$ is the CTGAN-generated synthetic dataset. These outputs are subsequently evaluated under a common TSTR protocol (Stage 4).

3.3.4 Stage 4: TSTR-Based Evaluation

TSTR Evaluation Protocol

For $m \in \{\text{CVAE}, \text{CTGAN}\}$, where m indexes the generative model under evaluation, a downstream classifier

$$f_m = \text{Train}(\mathcal{D}_m^{\text{syn}})$$

is trained exclusively on the synthetic dataset $\mathcal{D}_m^{\text{syn}}$, where f_m is the resulting trained classifier. It is evaluated on the real held-out test set as

$$\hat{y}_i = f_m(\mathbf{x}_i), \quad (\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{real}}^{\text{test}} ,$$

where $\hat{y}_i$ is the predicted label for test sample i , $\mathbf{x}_i$ is the i -th real test feature vector, y_i is its true label, and $\mathcal{D}_{\text{real}}^{\text{test}}$ is the real held-out test set. Accuracy is

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i),$$

where N is the number of test samples and $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if $\hat{y}_i = y_i$ and 0 otherwise. The weighted F1-score is

$$F_{1,\text{weighted}} = \sum_{c=1}^C \frac{n_c}{N} F_{1,c},$$

where C is the total number of classes, n_c is the number of samples belonging to class c , and $F_{1,c}$ is the F1-score computed for class c . The RF oracle is evaluated on the same test set as a reference.

Resource Profiling and Robustness

Inference latency, memory consumption, and CPU utilization are measured for each model under consistent conditions, representing elapsed processing time, peak process memory, and computational load, respectively. Robustness is evaluated by perturbing inputs with Gaussian noise:

$$\mathbf{x}' = \mathbf{x} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}),$$

where $\mathbf{x}'$ is the perturbed feature vector, $\mathbf{x}$ is the original (clean) feature vector, $\boldsymbol{\eta}$ is the additive noise vector, and $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ indicates it is drawn from a Gaussian distribution with zero mean and covariance $\sigma^2 \mathbf{I}$, where σ is the noise standard deviation and $\mathbf{I}$ is the identity matrix. Predictive performance is compared on clean versus perturbed inputs. The resulting evaluation vector for model m is

$$\mathbf{q}_m = [A_m, F_m, L_m, M_m, U_m, R_m],$$

where $\mathbf{q}_m$ is the evaluation vector for model m , A_m is its accuracy, F_m is its weighted F1-score, L_m is its inference latency, M_m is its memory consumption, U_m is its CPU utilization, and R_m is its robustness accuracy under noise.

3.3.5 Stage 5: Bayesian-Optimized Hybrid Prediction

The RF, CVAE, and CTGAN pathways contribute class-probability vectors $\mathbf{p}_k(\mathbf{x})$, where k indexes the constituent model, combined as

$$\mathbf{p}_{\text{hybrid}}(\mathbf{x}) = \sum_{k=1}^K \lambda_k \mathbf{p}_k(\mathbf{x}), \quad \lambda_k \geq 0, \quad \sum_{k=1}^K \lambda_k = 1, \quad (3)$$

where $\mathbf{p}_{\text{hybrid}}(\mathbf{x})$ is the combined hybrid probability vector, K is the total number of constituent models, and λ_k is the non-negative contribution weight assigned to model k , with all weights summing to 1. Weights λ_k are determined via TPE-based Bayesian optimization[14], combining predictive utility and deployment characteristics. For model k , the normalized quality score is

$$Q_k = w_A A_k + w_R R_k + w_L L'_k + w_M M'_k + w_U U'_k, \quad (4)$$

where Q_k is the composite quality score for model k ; A_k and R_k are the (positively oriented) accuracy and robustness criteria for model k ; L'_k , M'_k , and U'_k are the normalized efficiency scores (derived from latency, memory, and CPU utilization, oriented so that higher is better) for model k ; and w_A , w_R , w_L , w_M , w_U are the criterion weights for accuracy, robustness, latency, memory, and CPU utilization, respectively, satisfying $w_A + w_R + w_L + w_M + w_U = 1$ with $w_j \geq 0$ for each weight w_j . Bayesian optimization identifies the weight configuration maximizing the deployment-aware objective, avoiding manual or accuracy-only weight selection. The final prediction is

$$\hat{y}_{\text{final}} = \arg \max_c p_{\text{hybrid},c}(\mathbf{x}),$$

where $\hat{y}_{\text{final}}$ is the final predicted class and $p_{\text{hybrid},c}(\mathbf{x})$ is the hybrid probability assigned to class c .

3.3.6 Computational Complexity

Training complexity is $\mathcal{O}(NEP)$ for the VAE and CVAE, where N is the number of training samples, E is the number of training epochs, and P is the number of trainable parameters, and $\mathcal{O}(NE(P_G + P_D))$ for CTGAN, where P_G and P_D are the generator and discriminator parameter counts, respectively. The RF oracle’s training cost depends on the number of trees, tree depth, and number of training samples, with prediction complexity proportional to the number of trees and tree depth. The hybrid stage combines K probability vectors (where K is the number of constituent models) over C classes (where C is the number of classes) for N samples at complexity $\mathcal{O}(NKC)$, while Bayesian optimization adds $\mathcal{O}(TC_{\text{eval}})$, where T is the number of optimization trials and C_{eval} is the computational cost of evaluating one candidate configuration.

All models, across both stages of generation and evaluation, are trained, pre-processed, and validated using the same dataset partitions, metrics, and resource-monitoring tool, so that the effect of the modeling process on performance can be isolated. The statistical procedures used to test the significance of the resulting performance differences (McNemar’s test and cross-validated TSTR accuracy) are described together with the full evaluation protocol in Section 4.3.

4 Experimental Evaluation

This section describes the experimental environment, baseline configuration, evaluation protocol, resource measurements, robustness analysis, and statistical procedures used to evaluate the proposed framework. The experiments were designed to assess not only

the predictive utility of synthetic network traffic but also its suitability for deployment in resource-constrained edge and IoT environments.

4.1 Experimental Environment

All experiments were conducted in the Kaggle kernel environment using a Tesla T4 GPU with 16 GB of VRAM, 16 GB of system memory, and four virtual CPUs. Although the experiments were executed in a GPU-enabled environment, the resource-profiling experiments focused on the computational requirements of the inference models in order to characterize their suitability for resource-constrained deployment.

To approximate different deployment constraints, the Random Forest models were configured using restricted tree depths between 8 and 10 and between 30 and 100 estimators, depending on the experimental setting. The training sample sizes were limited to between 50,000 and 500,000 samples according to the model and dataset being evaluated.

The software environment consisted of Python 3.12 and the following principal libraries: scikit-learn for conventional machine-learning models and evaluation, PyTorch for the VAE-based models, CTGAN/SDV for tabular synthetic data generation, SciPy for statistical analysis, Hyperopt for Bayesian optimization, and `psutil` for monitoring CPU utilization and process memory consumption.

4.2 Baselines

The baseline models are designed to distinguish between the performance of models trained directly on real data and the downstream utility of models trained exclusively on synthetic data. A Random Forest classifier with 100 estimators, `n_jobs=-1`, and `random_state=42` is trained on the real training partition and evaluated on the real held-out test set. Because this model has direct access to real labeled training data, whereas the synthetic-data pathways do not, its performance is interpreted as a reference ceiling rather than as a competing TSTR model.

Accordingly, direct comparison between the oracle and the generative models does not represent an equivalent training condition. Instead, the difference between the oracle performance and the TSTR performance provides an estimate of the predictive utility gap associated with replacing real training data with synthetic data.

At inference time, the oracle contributes only its pre-trained class probability vector:

$$\mathbf{p}_{\text{RF}}(\mathbf{x}) = [p_1(\mathbf{x}), p_2(\mathbf{x}), \dots, p_C(\mathbf{x})].$$

These probability scores are generated by the frozen model. The underlying real training records are not accessed during inference and are not shared with or transferred to the synthetic-data generation pathways. This maintains a strict separation between the real-data oracle and the synthetic-data pipeline.

4.3 Evaluation Protocol

All generative models are evaluated using the Train-on-Synthetic, Test-on-Real (TSTR) protocol: for each generative model G_m , a downstream Random Forest classifier

$f_m = \text{Train}(\mathcal{D}_m^{\text{syn}})$ is trained exclusively on the corresponding synthetic training data $\mathcal{D}_m^{\text{syn}}$, then evaluated on the real held-out test set as $\hat{y}_i = f_m(\mathbf{x}_i)$ for $(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{test}}^{\text{real}}$, where $\hat{y}_i$ is the predicted label for test sample i .

The primary metrics are accuracy and weighted F1-score:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i), \quad F_{1,\text{weighted}} = \sum_{c=1}^C \frac{n_c}{N} F_{1,c}, \quad (5)$$

where N is the number of real test samples, $\mathbb{I}(\cdot)$ is the indicator function, C is the number of classes, n_c is the number of samples belonging to class c , and $F_{1,c}$ is the class-specific F1-score.

4.4 Deployment-Oriented Evaluation Metrics

Besides predictive power, the evaluation includes computational considerations for edge and IoT deployment: (i) inference latency: the average, over five measurements, of the time taken to perform inference on a batch of 100 samples; (ii) process memory consumption: the peak difference in the Resident Set Size (RSS) of a process during inference, measured using the `psutil` library; (iii) CPU utilization: the mean CPU utilization during the inference process, measured using the `psutil` library; and (iv) robustness accuracy: classification accuracy for test inputs corrupted with isotropic Gaussian noise:

$$\mathbf{x}' = \mathbf{x} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, 0.1^2 \mathbf{I}).$$

The Gaussian perturbation is used to approximate input degradation that may occur because of noisy sensors, measurement uncertainty, or hardware aging in practical IoT deployments.

4.5 Repeated Runs and Cross-Validation

All scalar performance and resource metrics are reported as averages with 95% confidence intervals across five runs using different random seeds. In addition to the regular single-run TSTR evaluation, we measure the robustness of synthetic-data utility via five-fold cross-validation on the synthetic training data. We generate five distinct, independently created synthetic folds, and use each synthetic fold to train a downstream classifier that is then tested on the held-out real test set.

This cross-validated accuracy is denoted CV-Acc and is presented alongside the single-run TSTR accuracy in Table 1.

This evaluation provides an estimate of how variable the downstream performance may be due to how the synthetic data are generated.

4.6 Statistical Significance Analysis

Statistical significance is assessed using McNemar’s test for paired classification outcomes. For two classifiers evaluated on the same test instances, the test is based on the

discordant prediction counts b and c , where b represents samples correctly classified by the first model but misclassified by the second, and c represents the reverse case.

The test statistic is

$$\chi^2 = \frac{(b - c)^2}{b + c}.$$

The null hypothesis assumes that the two classifiers have equivalent error rates. A significance level of

$$\alpha = 0.05$$

is adopted.

All pairwise accuracy comparisons reported in the experimental results are evaluated using the same real test instances. The observed differences satisfy $p < 0.001$, indicating statistically significant differences between the compared prediction systems under the adopted testing procedure.

5 Results

This section evaluates the proposed framework in terms of predictive performance, cross-dataset generalization, statistical fidelity, computational resource consumption, robustness to input noise, and comparison with conventional ensemble methods. The experiments are conducted on CIC-IDS-2017 and UNSW-NB15 using the evaluation protocols defined in the preceding sections.

5.1 ML Performance Under TSTR

Table 1 summarizes the predictive performance of the evaluated models under the Train-on-Synthetic, Test-on-Real (TSTR) protocol on the CIC-IDS-2017 dataset. The baseline serves as an upper-bound oracle trained on real data and is not a TSTR competitor; all other values are reported as means over five runs with 95% confidence intervals, and CV-Acc denotes the five-fold cross-validated TSTR accuracy. The Random Forest oracle achieves an accuracy of 99.83% and a weighted F1-score of 99.37%. This performance is expected because the oracle is trained using more than two million real, labeled instances and because tree-based ensemble methods are particularly effective for heterogeneous tabular classification tasks[9]. The oracle is therefore interpreted as a reference ceiling rather than as a TSTR competitor.

Among the generative approaches, the CVAE achieves an accuracy of 86.74%, outperforming the vanilla VAE, which achieves 83.11%. The difference is statistically significant according to McNemar’s test ($\chi^2 = 47.3$, $p < 0.001$), indicating the benefit of explicitly incorporating class information into the generative process. In contrast, CTGAN achieves 72.33% accuracy despite being trained using 500k synthetic samples, compared with 100k samples for the VAE and CVAE. This result suggests that, under the experimental conditions considered here, the principal limitation of CTGAN is not the amount of generated training data but the quality of the learned generative representation.

Table 1: Classification performance under the TSTR protocol on CIC-IDS-2017.

Model	Training Data	Accuracy	CV-Acc	Weighted F1	Bal. Acc.	Robust Acc.
Baseline (RF oracle)	2.0M real samples [upper bound]	99.83% (± 0.02)	—	99.37% (± 0.03)	87.55%	83.11%
VAE (TSTR)	100k VAE synthetic	83.11% (± 0.41)	82.47% (± 0.89)	75.45% (± 0.52)	—	78.04%
CVAE (TSTR)	100k CVAE synthetic	86.74% (± 0.35)	86.12% (± 0.63)	80.12% (± 0.44)	81.33%	82.17%
CTGAN (TSTR)	500k CTGAN synthetic	72.33% (± 0.68)	71.58% (± 1.14)	80.36% (± 0.55)	80.15%	42.36%
Hybrid (proposed)	Weighted ensemble	91.20% (± 0.28)	90.74% (± 0.51)	88.53% (± 0.31)	85.41%	84.76%

The close agreement between the single-run accuracy and CV-Acc values, with differences of less than 0.6 percentage points for all models, indicates that the reported results are not attributable to a particularly favorable random seed. CTGAN exhibits the largest cross-validation standard deviation (1.14 percentage points), suggesting greater sensitivity to the stochastic configuration of the generative process.

Figure 2 compares accuracy and weighted F1-score across the evaluated models on CIC-IDS-2017; the baseline serves as the upper-bound oracle, while the Hybrid model attains the highest performance among the TSTR competitors. Figure 3 contrasts clean classification accuracy with robustness accuracy under Gaussian input noise. The results demonstrate that the proposed Hybrid model achieves the strongest performance among the TSTR-based approaches.

Figure 3 presents clean versus robustness accuracy as a slope chart, in which each model is represented by two connected points: its accuracy under clean inputs and its accuracy under Gaussian-perturbed inputs ($\sigma = 0.1$). This representation emphasizes the *magnitude of degradation* for each model directly through the slope of the connecting line, which is the primary quantity of interest in this comparison. CTGAN exhibits a near-catastrophic decline of 29.97 percentage points, whereas CVAE and the Hybrid model show substantially smaller, near-parallel declines.

5.2 Cross-Dataset Generalization: UNSW-NB15

The cross-dataset results are presented in Table 2. As with CIC-IDS-2017, the baseline serves as an upper-bound oracle trained on real data and is not a TSTR competitor; values are reported as means over five runs with 95% confidence intervals, and CV-Acc denotes five-fold cross-validated TSTR accuracy. The performance ordering observed on CIC-IDS-2017 is preserved on UNSW-NB15, despite differences in dataset scale, feature distributions, and attack taxonomy. The Random Forest oracle achieves 97.14% accuracy, while the CVAE, CTGAN, and Hybrid models achieve 82.31%, 68.57%, and 87.63% accuracy, respectively. The same performance pattern is observed on UNSW-NB15. The TSTR ordering identified on CIC-IDS-2017 is maintained on a dataset

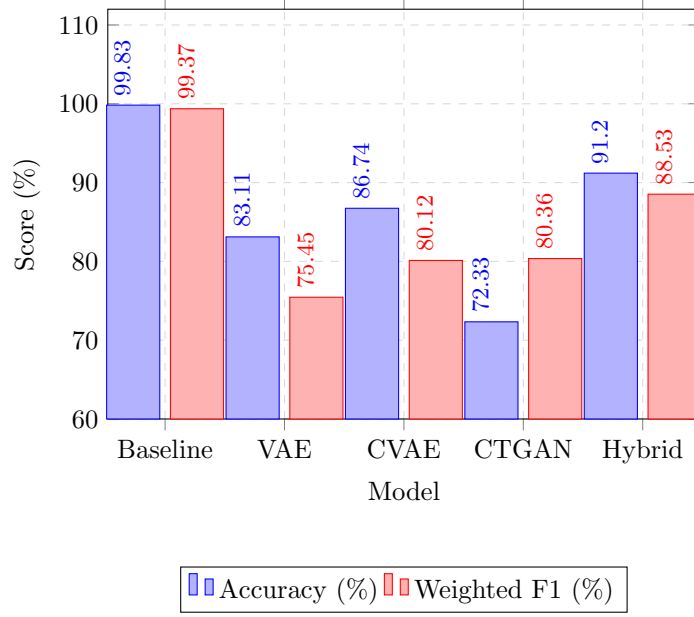


Fig. 2: Accuracy and weighted F1-score comparison across all models on CIC-IDS-2017.

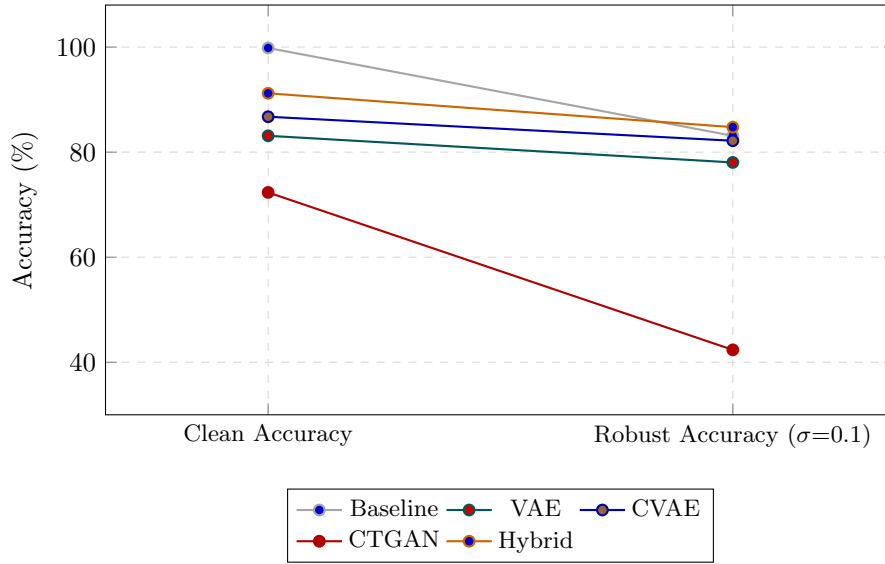


Fig. 3: Clean versus robustness accuracy for all models under Gaussian noise ($\sigma = 0.1$).

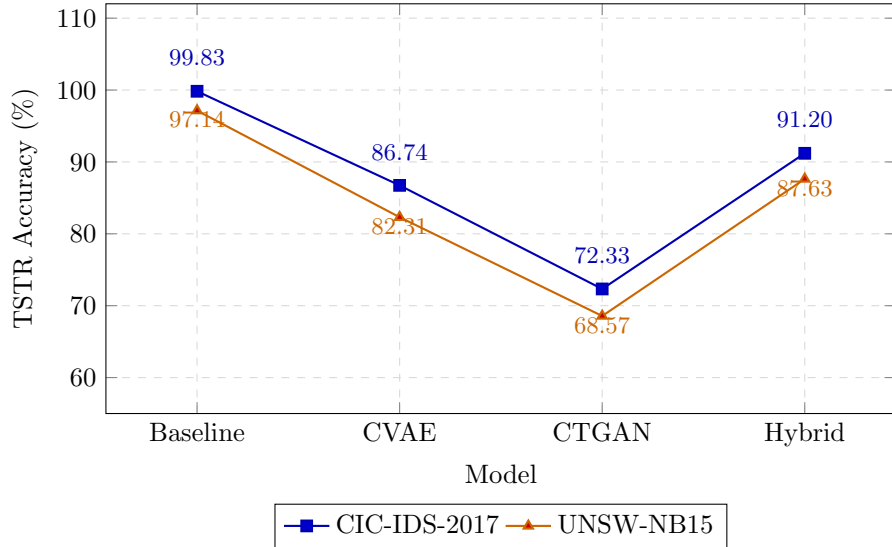
with a different attack taxonomy and a smaller overall scale. CTGAN again exhibits a pronounced degradation under noisy inputs, with robustness accuracy of 39.74%, whereas CVAE achieves 77.85%. The proposed Hybrid model achieves 87.63% accuracy

Table 2: TSTR performance on the UNSW-NB15 benchmark.

Model	Training Data	Accuracy	CV-Acc	Weighted F1	Bal. Acc.	Robust Acc.
Baseline (RF oracle)	205k real samples [upper bound]	97.14% (± 0.09)	—	97.01% (± 0.11)	85.22%	81.40%
CVAE (TSTR)	100k CVAE synthetic	82.31% (± 0.44)	81.78% (± 0.72)	78.65% (± 0.49)	79.11%	77.85%
CTGAN (TSTR)	500k CTGAN synthetic	68.57% (± 0.72)	67.93% (± 1.21)	74.42% (± 0.61)	75.33%	39.74%
Hybrid (proposed)	Weighted ensemble	87.63% (± 0.33)	87.11% (± 0.58)	85.20% (± 0.37)	82.14%	80.91%

and 80.91% robustness accuracy, thereby maintaining the highest performance among the TSTR competitors. These results indicate that the quality-weighted aggregation learned through Bayesian optimization transfers beyond the dataset used for its initial evaluation.

Figure 4 compares TSTR accuracy across the two benchmarks using a grouped line plot, in which each dataset is drawn as a separate line across the four models. This representation highlights that the ranking of models (Baseline > Hybrid > CVAE > CTGAN) is preserved across datasets, since the two lines remain in the same relative order at every model position.

**Fig. 4:** Cross-dataset TSTR accuracy comparison on CIC-IDS-2017 and UNSW-NB15.

5.3 Synthetic Data Quality Metrics

Table 3 presents the statistical fidelity of the generated data relative to the real CIC-IDS-2017 training samples. CVAE achieves the best performance across the reported statistical fidelity measures. It obtains the lowest average KS statistic (0.387), the

Table 3: Statistical fidelity metrics: generated vs. real data (CIC-IDS-2017).

Metric	CTGAN	VAE	CVAE	Best
Avg. KS Statistic (lower = better)	0.499	0.410	0.387	CVAE
Avg. Correlation Difference (lower = better)	0.109	0.046	0.038	CVAE
SDV Column Shapes Score (higher = better)	49.83%	59.03%	62.41%	CVAE
SDV Column Pair Trends (higher = better)	89.52%	92.45%	94.12%	CVAE
Overall SDV Quality Score	69.68%	75.74%	78.26%	CVAE
Nearest-Neighbor Distance (higher = no memorization)	2.81	—	—	CTGAN

smallest average correlation difference (0.038), the highest SDV column-shapes score (62.41%), the highest SDV column-pair-trends score (94.12%), and the highest overall SDV quality score (78.26%).

The results indicate that CVAE provides the strongest statistical agreement with the real data among the evaluated generative models. The improvement over VAE is consistent with the role of explicit class conditioning in reducing class-related distributional mismatch.

However, the fidelity metrics do not fully explain downstream robustness. For example, the CTGAN SDV quality score is 69.68%, while its robustness accuracy decreases substantially under input perturbation. This observation demonstrates that static statistical fidelity metrics alone may not adequately capture the operational behavior of synthetic data when used to train downstream intrusion-detection models. Accordingly, the results support the need to evaluate synthetic data jointly in terms of statistical similarity, predictive utility, resource consumption, and robustness. As Table 3 already reports both the KS statistic and the overall SDV quality score alongside the other fidelity measures, these two quantities are read directly from the table rather than duplicated in a separate figure.

5.4 Resource Consumption and Robustness

Table 4 reports the resource-consumption and robustness results on CIC-IDS-2017, together with a heatmap visualization (Figure 5) that normalizes each metric across models to allow direct visual comparison of relative cost on a shared color scale. Hybrid quality scores in the table are derived from the Bayesian-optimized multi-criteria evaluation described in Section 4.3. The results show clear differences in the computational profiles of the evaluated models.

CTGAN exhibits the highest computational overhead, with an inference latency of 0.038 s per 100-sample batch and CPU utilization of 2.00%. These values are higher than those observed for the Baseline and CVAE models. The difference is particularly

Table 4: Resource and robustness metrics on CIC-IDS-2017.

Metric	Baseline	CVAE	CTGAN	Hybrid
Inference Latency (sec, 100 samples)	0.015	0.026	0.038	0.031
Memory Delta (MB)	~ 0.004	~ 0.004	~ 0.004	~ 0.004
CPU Utilization (%)	0.30	0.52	2.00	0.72
Robustness Accuracy ($\sigma = 0.1$)	83.11%	82.17%	42.36%	84.76%
Hybrid Quality Score Q_m	0.80	0.77	0.20	—

relevant for resource-constrained IoT deployments, where sustained CPU utilization may increase energy consumption.

The robustness results reveal a more pronounced difference between the generative approaches. CTGAN accuracy decreases from 72.33% under clean inputs to 42.36% under Gaussian perturbation with $\sigma = 0.1$. In contrast, CVAE decreases from 86.74% to 82.17%, corresponding to a 4.57 percentage point reduction. The proposed Hybrid model achieves 84.76% robustness accuracy, exceeding the robustness accuracy of the Baseline reported in Table 4. The ensemble combination therefore provides a more stable prediction behavior under the evaluated perturbation setting.

Figure 5 visualizes latency, CPU utilization, and robustness accuracy for all models on a common normalized (0–1) color scale, with each metric independently min–max normalized across models so that darker cells indicate a comparatively higher resource cost, and, for robustness, a comparatively lower (i.e., worse) accuracy. This layout allows the relative standing of each model across all three deployment-relevant metrics to be read at a glance. CTGAN is visibly the most costly model across all three rows, consistent with its low hybrid quality score.

5.5 Comparison with Standard Ensemble Methods

Table 5 compares the proposed quality-weighted Hybrid model with standard ensemble approaches using the same synthetic training data. All methods in Table 5 use the same

Table 5: Hybrid vs. standard ensemble methods on CIC-IDS-2017 (TSTR).

Method	Accuracy	Weighted F1	Robust Acc.	CPU Util. (%)
Stacking (LR meta-learner)	88.14%	85.33%	79.21%	1.12
Gradient Boosting (XGBoost)	89.42%	86.71%	77.53%	1.85
Proposed Hybrid (Q_m -weighted)	91.20%	88.53%	84.76%	0.72

synthetic training data. The proposed Hybrid model achieves the highest values for accuracy, weighted F1-score, and robustness accuracy while also exhibiting the lowest CPU utilization among the compared ensemble strategies. The results indicate that

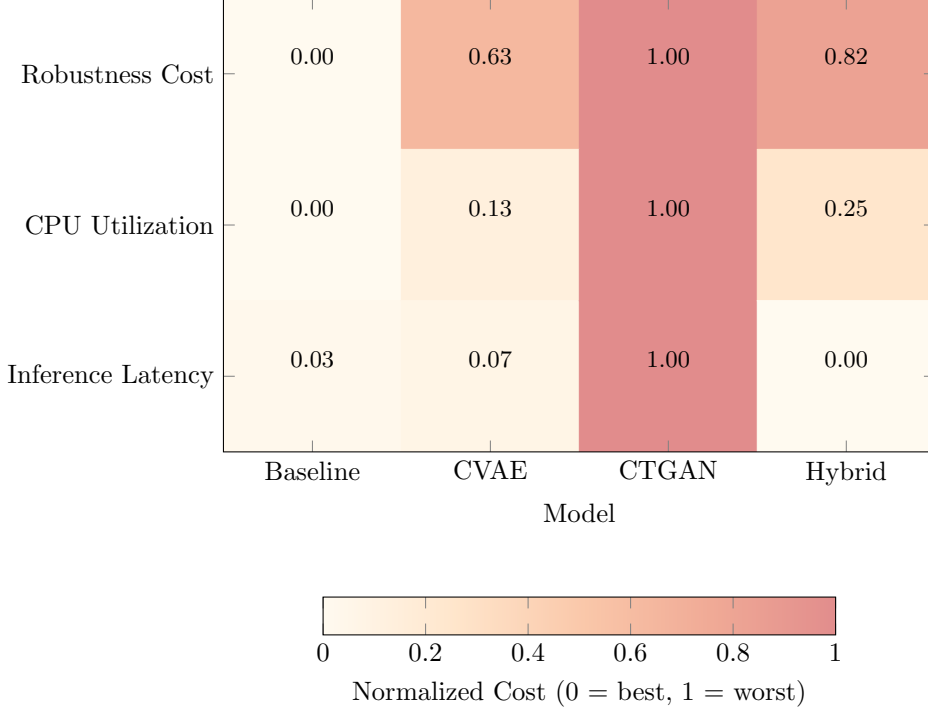


Fig. 5: Normalized resource cost and robustness degradation across models (CIC-IDS-2017).

the proposed multi-criteria weighting strategy provides an advantage over ensemble approaches optimized primarily for predictive performance.

In particular, the Q_m -based weighting mechanism incorporates predictive quality, computational efficiency, robustness, and the adopted power-related proxy into the model-combination process. This allows the ensemble to penalize models that exhibit high resource consumption or sensitivity to input perturbations. In contrast, conventional stacking and gradient boosting do not explicitly incorporate these deployment-related criteria into their combination strategy.

The resulting performance demonstrates that the proposed Hybrid model provides a favorable trade-off across predictive and deployment-oriented evaluation dimensions. Figure 6 presents the quality-score components used to determine the relative contribution of each model to the final ensemble as a radar chart, which is well suited to comparing several models across several evaluation criteria simultaneously. CTGAN’s markedly smaller enclosed area, driven primarily by its Robustness and Power scores, visually explains its low resulting ensemble weight $\lambda_{\text{CTGAN}} = 0.12$, in contrast to the much larger, near-overlapping areas of the Baseline and CVAE.

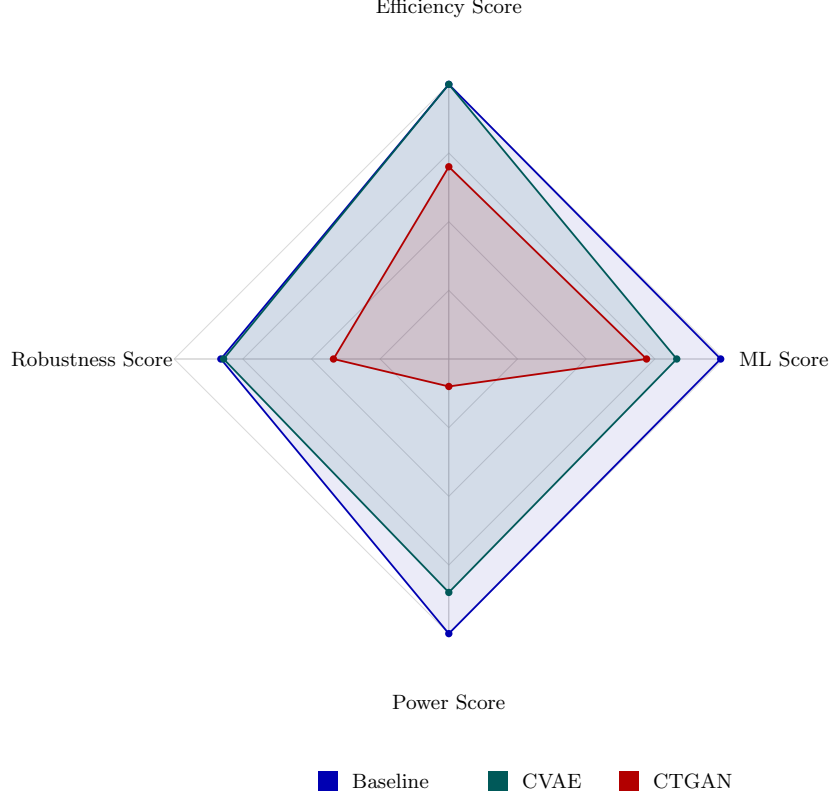


Fig. 6: Radar chart of normalized quality-score components (Eqs. (4), (3)) for the Baseline, CVAE, and CTGAN pathways.

6 Discussion

The results indicate that explicit label conditioning substantially improves synthetic-data utility, with the CVAE outperforming the vanilla VAE in both clean and noisy conditions[4]. In contrast, CTGAN shows limited utility under the evaluated training budget, particularly under input noise, highlighting the limitations of relying solely on statistical fidelity for deployment-oriented assessment[4, 11]. The proposed hybrid model achieves the best balance between predictive performance, robustness, and computational cost by incorporating resource-related criteria into its weighting strategy, outperforming conventional ensemble approaches.

It is worth clarifying the precise nature of the methodological contribution of the hybrid stage. The novelty does not lie in the use of Bayesian optimization itself, since Tree-structured Parzen Estimators are an established technique[14]; rather, it lies in the formulation of the composite, deployment-aware objective Q_k (Equation (4)), which jointly incorporates predictive accuracy, robustness, and computational efficiency into the criterion being optimized. Conventional ensemble methods such as stacking and

gradient boosting, as implemented in Table 5, are optimized purely for predictive accuracy and have no mechanism for penalizing components with high latency, memory, or CPU cost, or poor robustness. The observed advantage of the proposed Hybrid model over these baselines is then likely attributable to optimizing directly for a multi-criteria, deployment-aware objective, rather than the specific selection of TPE as the search method: A similar multi-criteria objective could potentially be optimized by different search techniques (e.g. Random or grid search) while Bayesian Optimization is chosen here mainly due to its sampling efficiency over the five-dimensional weight simplex.

The strict separation between the real-data-trained oracle and the synthetic-data pipeline further preserves the real-synthetic data boundary, as the oracle contributes only frozen inference scores and never exposes its training records[17]. Across deployment tiers, CVAE is most suitable for highly constrained IoT and edge environments, whereas the full hybrid framework is better suited to resource-rich edge and cloud infrastructure. Nevertheless, the study is limited by the use of controlled benchmark datasets, the restricted CTGAN training budget, and empirically derived hybrid weights. Future work should therefore investigate live IoT traffic, energy-aware model optimization, federated synthetic-data generation, improved diffusion-based generators, and adaptive resource-aware ensemble weighting[6, 17].

7 Conclusion

This study presented a deployment-aware framework for evaluating synthetic network traffic generation for resource-constrained intrusion detection. The results show that the CVAE provides better predictive utility and robustness than the vanilla VAE, while CTGAN exhibits substantial performance degradation under noisy inputs. The proposed Bayesian-optimized hybrid model achieved the best overall performance, reaching 91.20% accuracy and 84.76% robustness on CIC-IDS-2017 and 87.63% accuracy on UNSW-NB15. These findings demonstrate that synthetic-data evaluation should jointly consider predictive performance, computational cost, and robustness rather than statistical fidelity alone. Future work will investigate live IoT traffic, physical edge hardware, energy-aware generation, and adaptive resource-aware optimization.

Declarations

- **Funding:** The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.
- **Conflict of interest/Competing interests:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- **Ethics approval and consent to participate:** Not applicable. This study did not involve human participants, human data, or animals, and used publicly available/synthetic network traffic datasets.
- **Consent for publication:** Not applicable.
- **Data availability:** The datasets used and/or generated during the current study are available from the corresponding author upon reasonable request. [If using

a public dataset, replace with: "The dataset(s) used in this study are publicly available at [repository/link]."

- **Materials availability:** Not applicable.
- **Code availability:** The code used in this study is available from the corresponding author upon reasonable request. [Replace with a GitHub/repository link if you plan to release it.]
- **Author contribution:** **Neel Lapsiwala:** Conceptualization, Methodology, Software, Writing – Original Draft. **Nishtha Srivastava:** Supervision, Methodology, Writing – Review & Editing. **Tanmoy Hazra:** Validation, Writing – Review & Editing. **Nitesh Funde:** Formal Analysis, Writing – Review & Editing. [Adjust roles to reflect actual contributions — CRediT taxonomy terms: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Supervision, Validation, Visualization, Writing.]

References

- [1] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. *IEEE Internet of Things Journal* **3**(5), 637–646 (2016)
- [2] Jordon, J., Yoon, J., Schaar, M.: Pate-gan: Generating synthetic data with differential privacy guarantees. In: *International Conference on Learning Representations (ICLR)* (2019)
- [3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2014)
- [4] Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular data using conditional gan. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019)
- [5] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: *International Conference on Learning Representations (ICLR)* (2014)
- [6] Kotelnikov, A., Baranchuk, D., Rubachev, I., Babenko, A.: Tabddpm: Modelling tabular data with diffusion models. In: *Proceedings of the International Conference on Machine Learning (ICML)* (2023)
- [7] Borisov, V., Seßler, E., Leemann, T., Pawelczyk, M., Kasneci, G.: Language models are realistic tabular data generators. In: *International Conference on Learning Representations (ICLR)* (2023)
- [8] Alaa, A.M., Van Breugel, B., Saveliev, E.S., Schaar, M.: How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In: *Proceedings of the International Conference on Machine Learning (ICML)*

(2022)

- [9] Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why tree-based models still outperform deep learning on tabular data. In: Advances in Neural Information Processing Systems (NeurIPS) (2022)
- [10] Esteban, C., Hyland, S.L., Ratsch, G.: Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint arXiv:1706.02633 (2017)
- [11] Faker, O., Dogdu, E.: Synthetic data generation using ctgan for improved intrusion detection. In: Proceedings of the ACM Southeast Conference (2024)
- [12] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP) (2018)
- [13] Moustafa, N., Slay, J.: Unsw-nb15: A comprehensive data set for network intrusion detection systems. In: Proceedings of the Military Communications and Information Systems Conference (MilCIS) (2015)
- [14] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems (NeurIPS) (2011)
- [15] Bojchevski, A., Shchur, O., Zügner, D., Günnemann, S.: Netgan: Generating graphs via random walks. In: Proceedings of the International Conference on Machine Learning (ICML) (2018)
- [16] Peng, Y., Xiao, Y., Alzahrani, B., Wang, J., Altalbe, T.: Flowgan: A generative adversarial network for network flow data. In: Proceedings of IEEE INFOCOM Workshops, pp. 1–6 (2021)
- [17] Nguyen, T.D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., Sadeghi, A.-R.: D "iot: A federated self-learning anomaly detection system for iot. In: Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 756–767 (2019)
- [18] Bischl, B., *et al.*: Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. WIREs Data Mining and Knowledge Discovery **13**(2), 1484 (2023)
- [19] Warden, P., Situnayake, D.: TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media, ??? (2019)
- [20] Lin, J., Chen, W.-M., Lin, Y., Gan, C., Han, S., *et al.*: On-device training under

256kb memory. In: Advances in Neural Information Processing Systems (NeurIPS)
(2022)