Cost-Aware Adversarial Robustness Evaluation for ML-Based Intrusion Detection in 5G Network Slicing

João Vitor da Silva Campos, Abdel Fadyl Chabi, Andson Marreiros Balieiro
Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE)
Recife, Pernambuco, Brazil
{jvsc2, afc9, amb4}@cin.ufpe.br

***Abstract*—This paper proposes a structured five-stage evaluation pipeline to analyze the trade-offs among detection accuracy, adversarial robustness, and computational efficiency in machine learning-based traffic classification for 5G network slicing. The pipeline is applied to three representative models, a Multi-Layer Perceptron (MLP), an XGBoost classifier, and a supervised Autoencoder (AE), using the 5G-SliciNDD dataset. Feature selection based on Mutual Information (MI) and Principal Component Analysis (PCA) reduced the original feature set by more than 50% while maintaining a classification accuracy above 0.99. Statistical validation using the Friedman and Nemenyi tests confirmed XGBoost as the best-performing model under clean conditions. Under adversarial evaluation, MLP and AE experienced substantial performance degradation under Projected Gradient Descent (PGD) attacks, whereas XGBoost required computationally expensive decision-based attacks to achieve comparable degradation. Although adversarial training significantly improved the robustness of the neural models, it incurred considerable computational and energy overhead. In contrast, XGBoost achieved partial robustness recovery with only a 3% retraining energy increase, excluding the one-time offline cost of generating the reused adversarial examples. These findings highlight the importance of jointly evaluating predictive performance, adversarial robustness, and computational efficiency when selecting intrusion detection models for resource-constrained 5G network slicing environments.**

***Index Terms*—Machine Learning; 5G Network Slicing; Adversarial Robustness; Computational Efficiency.**

I. Introduction

The Fifth Generation of Mobile Networks (5G) enables ultra-low latency, high throughput, and massive connectivity across heterogeneous service classes, such as Ultra-Reliable Low-Latency Communications (URLLC), enhanced Mobile Broadband (eMBB), and massive Machine-Type Communications (mMTC) [1] through network slicing (NS), which partitions compute, storage, and network resources into isolated logical networks sharing the same physical infrastructure [2]. Machine learning (ML) is widely used to automate traffic classification in this setting [3], [4], but the co-existence of multiple slices over shared infrastructure expands the attack surface, making classifiers targets for adversarial manipulation [5]. For instance, imperceptible input perturbations [6] can cause a malicious flow to be misclassified as benign, evading the intrusion detection mechanisms that are relied upon to safeguard slice isolation. Critically, differentiable model-based solutions (e.g., neural networks) expose explicit gradients to the attacker, whereas non-differentiable models (e.g., gradient-boosted trees) require more computationally expensive decision based black-box attacks. Therefore, this distinction should be assessed alongside the computational efficiency requirements of resource-constrained edge networks, considering the overhead introduced by defense mechanisms.

Prior studies have advanced adversarial robustness evaluation in Intrusion Detection Systems (IDS), but with narrower scope. For instance, Ahmadinejad et al. [3] achieved over 99% accuracy in 5G slice classification, evaluated only under clean conditions. Alslman et al. [11] extend [3] using the same dataset (5G-SliciNDD) under six Generative Adversarial Network (GAN)-based attacks on Artificial Neural Network (ANN), Decision Tree (DT), and Random Forest (RF) classifiers with a CycleGAN-based recovery defense. However, it does not consider either gradient-based or decision-based attack detection nor does it assess the resource consumption. Dang et al. [7] found XGBoost to be more resilient than neural networks under Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) attacks. The authors applied gradient-based attacks uniformly to both differentiable and non-differentiable models, overlooking the mismatch between attack methodology and model architecture. This concern was also raised by Parikh et al. [8], although without empirical validation. Beyond IDS, Shi et al. [14] is the only work that investigates adversarial robustness in network slicing. However, it focuses on radio resource allocation rather than traffic classification.

In this respect, there is a lack of studies that jointly evaluate differentiable and non-differentiable models under architecture-appropriate white-box/black-box threats using a 5G slicing dataset and characterize the resulting computational and energy costs. This paper addresses these gaps by proposing a five-stage pipeline applied to three ML model architectures: fully differentiable neural networks (Multi-Layer Perceptron, MLP), non-differentiable tree ensembles (XGBoost), and a reconstruction-based architecture (Autoencoder, AE), for binary traffic classification in 5G slicing. This proposal enables a direct study of how architecture shapes both vulnerability and defense cost, addressing the mismatch observed in [7], [8].

The proposed pipeline comprises data preprocessing, feature selection using Mutual Information (MI) and Principal Component Analysis (PCA), statistically validated hyperparameter optimization, architecture-appropriate attacks (white-box for MLP/AE, black-box for XGBoost), and defense assessment, measuring detection accuracy, Area Under the Curve (AUC), robustness, and computational and energy cost in terms of Central Processing Unit (CPU) utilization, memory consumption, execution time, energy consumption. The results show that AVG-based feature selection reduces the original feature set by more than 50% without degrading performance, that XGBoost is statistically superior under clean conditions and imposes a substantially higher attack cost on an adversary than MLP/AE-based models, and that adversarial training enhances robustness at the expense of higher energy consumption.

This paper is organized as follows. Section II describes the proposed methodology. Section III discusses the experimental results. Section IV concludes this paper and outlines future research directions.

II. Methodology

The proposed pipeline follows the methodology shown in Fig. 1, which comprises five main stages, described as follows.

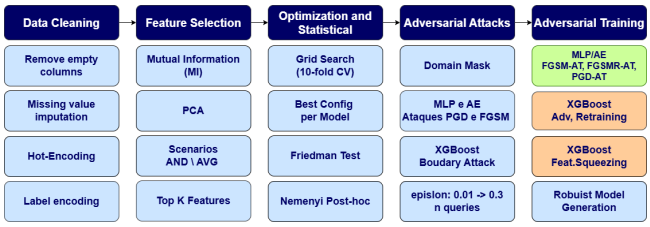


Fig. 1. Proposed methodological pipeline illustrating the five main stages.

A. Data Cleaning (Preprocessing)

The 5G-SliciNDD dataset [9] is used in this work. It is a publicly available dataset comprising 14,456 network flow records described by 74 features, representing global 5G traffic and including traffic statistics and labels associated with both benign and malicious activities.

To prepare the dataset for feature selection, model training, and evaluation, a preprocessing pipeline consisting of three steps was applied. First, non-informative attributes, including index-like columns and features with excessive missing values, were removed. Second, columns with some missing values were filled according to the semantics of each feature. Specifically, Transmission Control Protocol (TCP)-specific attributes were assigned a value of -1 to represent non-TCP flows, while the remaining numerical features were imputed using median of their respective columns. Third, the target variable was encoded as 0 (Benign) and 1 (Malicious). After preprocessing, the dataset contained no missing values.

B. Feature Selection

The second stage aimed to reduce dimensionality while preserving discriminative power and minimizing computational cost. Two independent feature-ranking strategies were adopted: Mutual Information (MI), which quantifies the statistical dependence between each feature and the target class, and Principal Component Analysis (PCA), which projects the data onto orthogonal components that maximize the explained variance [16]. Both methods were implemented using the Scikit-learn library [10].

The PCA score of feature j was computed on standardized data as $\mathrm{PCA}_j = \sum_i \rho_i |v_{ij}|$, where v_{ij} is its loading on principal component i and ρ_i is the explained-variance ratio of that component. Features were discarded using the thresholds $MI < 0.1$ and $PCA < 0.002$, set after observing a sharp drop in the scores beyond these points. The remaining features were organized into two strategies: AND, which retains only the features selected by both MI and PCA, and AVG, which ranks those selected by at least one of them by the average of their normalized MI and PCA ranks and retains the top K. The models were trained with K increasing from 5 in steps of 5, up to the number of features available, to identify the point beyond which additional features no longer improve performance. Only 32 features satisfy both thresholds, so the AND curves in Fig. 2 end at $K = 32$.

C. Hyperparameter Optimization and Statistical Validation

To mitigate overfitting to a single train/test split, the models' hyperparameters were optimized through an exhaustive grid search evaluated using stratified 10-fold cross-validation, accelerated by a successive-halving strategy, which progressively reduces the set of candidate configurations based on partial evaluations. For each model, the hyperparameter configuration yielding the highest mean AUC across the stratified 10-fold cross-validation was selected as the final configuration, resulting in a MLP with 128 hidden units, dropout of 0.3, learning rate of 5e-4, batch size of 128, and 50 epochs; a XGBoost with a maximum tree depth of 7, learning rate of 0.1, 300 estimators, and subsample ratio of 1.0; and an AE with encoding dimension of 32, 64 hidden units, dropout of 0.3, learning rate of 1e-3, batch size of 128, and 50 epochs.

To determine whether the observed performance differences between the three best configurations were statistically significant rather than an artifact of a particular data split, the corresponding per-fold AUC scores were compared using the Friedman test [15], a non-parametric test that ranks the models within each fold and evaluates whether the resulting rank distribution deviates from the null hypothesis of equal performance across models. Because a significant Friedman result only indicates that at least one model differs from the others, it was followed by the Nemenyi post-hoc test [15], which performs all pairwise comparisons while controlling the family-wise error rate ($\alpha = 0.05$).

D. Adversarial Attacks

After optimization, adversarial robustness was evaluated using attack methods according to the model architectures. All attacks were applied to the subset of features selected in the feature selection stage. Before generating attacks, a feature-level domain mask was built from the training set to ensure that

adversarial perturbations remain semantically valid. For each feature, the mask encodes: a binary perturbability indicator that prevents modification of features such as binary identifiers and protocol indicators; minimum and maximum limits per feature derived from the training distribution; and index lists that identify features with integer and binary values, which require post-perturbation rounding and clipping to preserve domain consistency. Perturbations were calculated in the scaled feature space to ensure numerical stability during gradient calculation and subsequently projected back to the raw feature space via the inverse transformation of the tuned Standard Scaler, ensuring that the generated adversarial samples remain consistent with the constraints of the network traffic domain.

The MLP and AE models were submitted to white-box gradient-based attacks. Since both MLP and AE are end-to-end differentiable, exposing explicit gradients to the attacker, the PGD attack was chosen. The PGD operates by iteratively maximizing the model loss with respect to the input features, applying successive gradient steps of size α and projecting the result back into a permissible neighborhood ε around the original sample after each iteration, as defined in Eq. 1, where $x^{(t)}$ denotes the adversarial sample at iteration t, $\alpha = 0.2 \cdot \varepsilon$ is the step size, $\mathcal{L}$ is the model loss function, θ represents the model parameters, and clip_ε ensures the total perturbation remains bounded by ε. In this stage, the AE and MLP models were subjected to attacks where ε varied throughout. [0.01, 0.05, 0.10, 0.20, 0.30].

$$x^{(t+1)} = \text{clip}_\varepsilon \left(x^{(t)} + \alpha \cdot \text{sign} \left(\nabla_x \mathcal{L}(\theta, x^{(t)}, y) \right) \right) \quad (1)$$

XGBoost is a non-differentiable model, which makes gradient-based attacks inapplicable. Therefore, the Boundary Attack [17] was implemented: a decision-based black-box attack that relies only on hard-label predictions, performing a random walk along the decision boundary that iteratively shrinks the perturbation via binary search, which is why it requires far more queries than a gradient-based attack.

This attack was implemented using the Adversarial Robustness Toolbox (ART) [18] and executed against an encapsulated XGBoost classifier, exposed through the ART's BlackBoxClassifier interface, which restricts the attacker to only prediction queries, without access to the model's internal details or gradients. It was evaluated with maximum iteration budgets [5, 10, 20, 30, 50, 75, 100]. Given that the cost to the attacker in black-box configurations is more naturally measured by the number of queries to the model than by the magnitude of the disruption, we additionally report the total number of queries (n_queries) required per attack configuration.

E. Adversarial Defenses

For MLP and AE, three adversarial training (AT) strategies were evaluated: standard Fast Gradient Sign Method adversarial training (FGSM-AT), Projected Gradient Descent adversarial training (PGD-AT), and FGSM restart adversarial training (FGSMR-AT).Each AT strategy incorporates adversarial examples directly into training: during each epoch, adversarial counterparts are generated online for each clean mini-batch using the corresponding attack method and domain mask, and are used immediately in the gradient update without being stored. A fixed ratio of 50% adversarial samples per batch was applied in all variants. Training was performed with a fixed perturbation magnitude of ε = 0.10, and the resulting models were evaluated across the entire range of ε values used in Section II-D.

For XGBoost, two defense strategies were considered: Adversarial Retraining (AdvRetraining) and Feature Squeezing. The former augments the training set with adversarial examples generated using the Boundary Attack and retrains the model on the resulting combined dataset. Unlike online generation of adversarial examples for model training, AdvRetraining generates the adversarial examples offline, caching them for retraining process. The latter reduces the model's sensitivity to adversarial perturbations by applying Gaussian Input Smoothing ($\sigma \in \{0.005, 0.01, 0.02, 0.05\}$) to the input features during inference, mitigating subtle adversarial perturbations without modifying the trained model.

F. Metrics and Energy Consumption Estimation

The solutions were analyzed in terms of detection accuracy, AUC, CPU utilization, memory consumption, execution time, and energy consumption. The last metric was estimated according to [12], which relies on average CPU utilization and the processor's nominal Thermal Design Power (TDP) as an approximation for full-load power draw (a thermal specification, not a direct power measurement). The experiments ran on an AMD Ryzen 5 5500X3D processor (6 cores, 12 threads) with a manufacturer-rated TDP of 105 W [13].

The CPU utilization was collected with `psutil`, which reports aggregated occupancy across the processor's 12 logical cores (up to 1200% at full load). The average utilization was normalized by this theoretical maximum to obtain the estimated power P (Eq. 2), which was then multiplied by the execution time t to obtain the estimated energy E.

$$P = \frac{CPU_{mean}}{1200} \cdot 105 \qquad E = P \cdot t \quad (2)$$

III. Results

All experiments were conducted on the machine described in Section II-F, with 16 GB of DDR4 RAM (2×8 GB at 3200 MHz), running within a Jupyter Notebook environment under Miniconda. Each pipeline stage was executed in an isolated subprocess to ensure unbiased resource measurement, and CPU utilization, peak memory usage (Resident Set Size, RSS), and wall-clock time were collected by a background thread using `psutil`, sampling at fixed intervals.

A. Data Preprocessing and Feature Selection

During preprocessing, two features with approximately 95% missing values were discarded. Fig. 2 presents the accuracy achieved by the models as a function of the number of selected features considering the AVG and AND feature selection strategies. The AVG strategy outperformed the AND strategy

across all models, indicating that combining the MI and PCA rankings yields a more discriminative feature ranking than retaining only the features selected by both methods. All three models exhibited a rapid increase in accuracy for small values of k, suggesting that the most informative features are concentrated among the highest-ranked features in the AVG strategy. XGBoost reached near-peak accuracy at $k = 20$ ($Acc = 0.995$), whereas MLP and AE required slightly more features to stabilize, converging at approximately $k = 25$–30. To ensure a fair and consistent comparison throughout the evaluation pipeline, a common setting of $k = 35$ under the AVG strategy was adopted for all models.

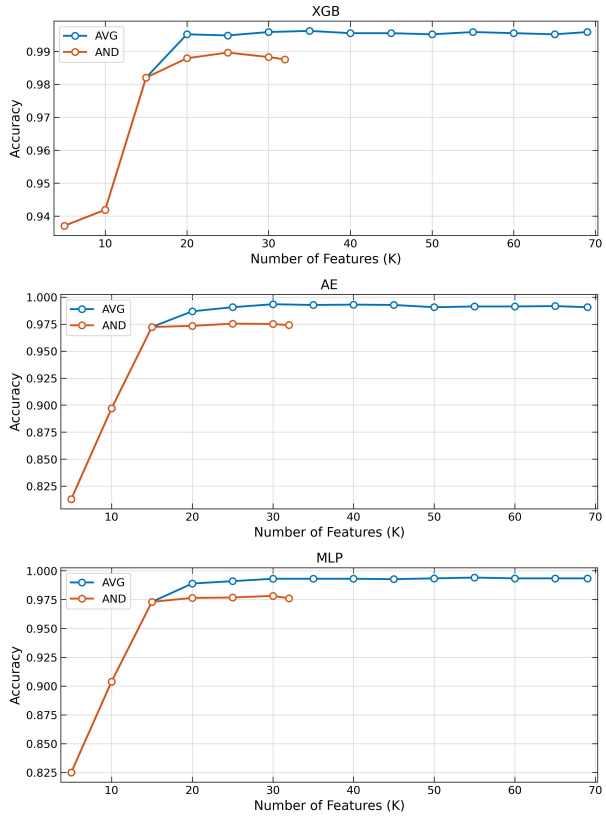


Fig. 2. Accuracy as a function of the number of selected features (K) under the AVG and AND scenarios for XGBoost, AE, and MLP.

Regarding computational cost during feature selection, MLP and AE exhibited stable CPU usage (145–205%), with memory growing monotonically with k (366–388 MB). XGBoost exhibited a larger CPU footprint (700–1020%), reflecting its parallel tree construction, while its memory (328–343 MB) also increased with k but remained lower than that of the neural networks.

B. Statistical Validation of Model Comparison

Table I reports the cross-validated AUC (mean ± std over 10 stratified folds) for the best configuration of each model. The Friedman test rejected the null hypothesis of equal performance among the models ($\chi^2 = 15.80$, $p = 0.0004$). The subsequent Nemenyi post-hoc test revealed that XGBoost differs significantly from both MLP ($p = 0.0102$) and AE ($p = 0.0004$), whereas no statistically significant difference was observed between MLP and AE ($p = 0.6438$). These findings are consistent with the average model ranks across the cross-validation folds (XGBoost: 1.0; MLP: 2.3; AE: 2.7). Although statistically significant, the gap is practically small (mean AUC of 0.9998 versus 0.9992): the test shows that XGBoost's advantage is consistent across folds rather than large, so clean-data performance alone barely separates the models, and the differences that matter emerge under attack and in defense cost (Sections III-C and III-D).

TABLE I
CROSS-VALIDATED AUC (10 FOLDS) PER MODEL.

Model	AUC (Mean ± Std.)
MLP	0.9992 ± 0.0008
XGBoost	0.9998 ± 0.0001
AE	0.9992 ± 0.0007

C. Adversarial Attacks

Fig. 3 shows the accuracy of the MLP and AE classifiers under PGD attack at different perturbation magnitudes. Both models maintain strong performance at low perturbation levels, with accuracy above 0.97 for $\varepsilon \leq 0.05$. As ε increases, performance degrades sharply: at $\varepsilon = 0.20$, the MLP reaches an accuracy of 0.519 whereas the AE of 0.525, both close to the random-hit level. With $\varepsilon = 0.30$, accuracy drops further to 0.434 and 0.438, respectively, with evasion rates exceeding 96% for MLP and 95% for AE. The similar profiles of MLP and AE stem from a shared property: both are end-to-end differentiable, so the AE's compression stage is not a gradient barrier and gradients flow through its encoder and decoder just as in the MLP.

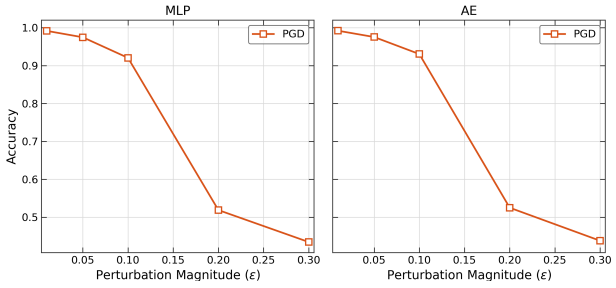


Fig. 3. Accuracy under PGD attack for MLP and AE across perturbation magnitudes (ε).

Fig. 4 illustrates the accuracy of the XGBoost classifier under the Boundary Attack as a function of the total number of model queries. The Boundary Attack achieves its most significant degradation with only 451,700 queries (5 iterations), reducing accuracy from 0.996 to 0.500, an evasion rate of 97.6%. Increasing the query budget beyond this point does not produce further degradation: accuracy partially recovers to approximately 0.61 at 1.3 million queries and stabilizes around 0.58–0.60 for budgets up to 5.7 million queries. This is a known property of the attack, not an artifact: its random walk stays on the misclassified side of the boundary while

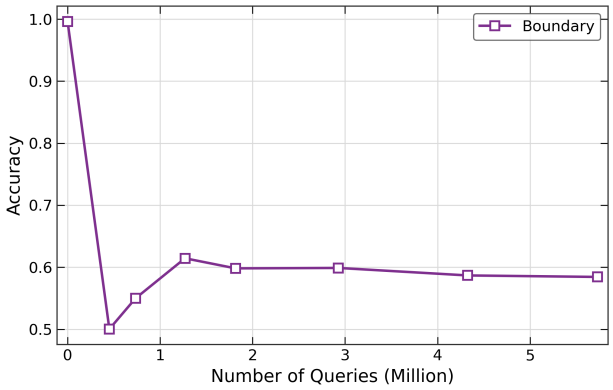


Fig. 4. Accuracy under Boundary Attack as a function of the query budget (`n_queries`) for XGBoost.

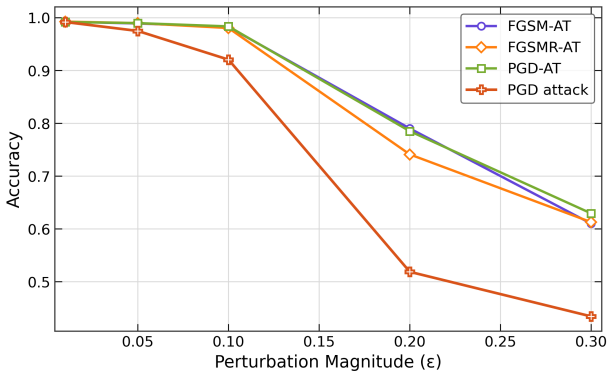


Fig. 5. Accuracy under adversarial training defenses against PGD attack (MLP).

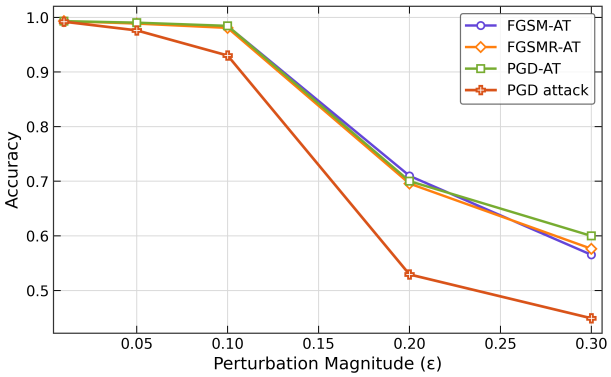


Fig. 6. Accuracy under adversarial training defenses against PGD attack (AE).

minimizing perturbation, so accuracy need not fall monotonically once no lower-perturbation direction remains, indicating a higher query cost to fully collapse the boundary rather than indefinite resistance (Table II).

Table II reports the computational and energy cost of attack generation for each model. Attack generation under PGD is essentially free for MLP and AE, requiring under 0.2 s and 2.2 J even at the strongest perturbation tested ($\varepsilon = 0.30$), consistent with their fully differentiable, gradient-accessible structure. XGBoost, under the Boundary Attack, is several orders of magnitude more expensive, requiring approximately 2,500–2,800$\times$ more time and 18,000–26,000$\times$ more energy than a PGD step on MLP or AE. These results confirm that breaking XGBoost under the evaluated attacks, largely because of its architectural opacity, imposes a disproportionately higher energy cost on attackers operating under constrained computational or power budgets.

TABLE II
COMPUTATIONAL AND ENERGY COST OF THE ATTACKS USED.

Model	Attack	Time (s)	CPU (%)	Energy (J)
MLP	PGD	0.142	124.40	1.55
AE	PGD	0.162	155.11	2.20
XGBoost	Boundary	397.81	1155.54	40222.5

D. Adversarial Defenses

Figs. 5 and 6 show the effectiveness of adversarial training strategies against PGD attacks for the MLP and AE models. All three defenses substantially improve robustness compared to the undefended models, particularly for moderate and high perturbation levels, where the original classifiers exhibited a rapid degradation in accuracy. Among the evaluated approaches, PGD-AT consistently achieves the highest accuracy at larger perturbation magnitudes, which is expected since the defense is optimized using the same iterative attack employed during evaluation. FGSM-AT produces comparable results despite relying on a single-step adversarial generation process, indicating that much of the robustness gain can be obtained with a considerably lower training complexity. FGSMR-AT, while consistently outperforming the undefended models, exhibits a slightly larger accuracy degradation for stronger attacks, suggesting that its regularization strategy is less effective at modeling the iterative perturbation process characteristic of PGD.

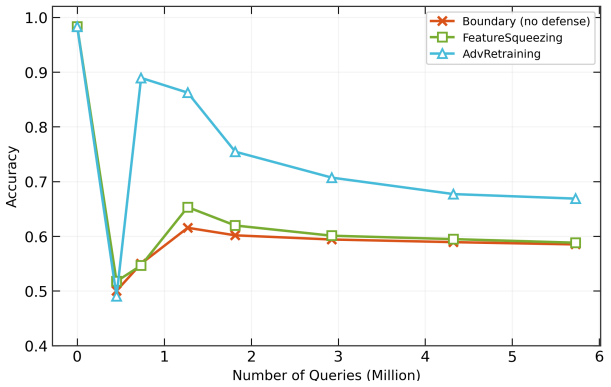


Fig. 7. Accuracy under AdvRetraining and Feature Squeezing against Boundary Attack as a function of the query budget (XGBoost).

Fig. 7 shows that the two defense strategies behave differently against the Boundary Attack. Feature Squeezing provides virtually no robustness improvement, as its accuracy closely follows the undefended model across the entire query budget. This is expected because Boundary Attack is decision-based and therefore little affected by the input quantization introduced by Feature Squeezing. In contrast, AdvRetraining consistently improves robustness, increasing accuracy to ap-

proximately 0.67–0.71 and maintaining this advantage even for large query budgets. Although clean-data performance is not fully recovered, retraining effectively shifts the decision boundary, making adversarial examples harder to find. As shown in Table III, this robustness gain is achieved with only a 3% increase in execution time and energy consumption, whereas adversarial training for MLP and AE increases execution time by up to $64\times$ and energy consumption by more than 6300%. This asymmetry compares different quantities: MLP/AE overhead already embeds online, per-epoch generation cost, whereas AdvRetraining reuses examples generated once offline, whose cost (397.81 s, $40{,}222.5$ J, Table II) is not amortized into the 3% figure.

TABLE III
COMPUTATIONAL AND ENERGY OVERHEAD OF DEFENSE STRATEGIES.

Model	Defense	Time (s)	Energy (J)	Time Overhead	Energy Overhead
MLP	FGSM-AT	56.34	557	$13.7\times$	+1267%
MLP	FGSMR-AT	66.24	640	$16.1\times$	+1508%
MLP	PGD-AT	232.62	2054	$56.5\times$	+5546%
AE	FGSM-AT	268.09	2321	$47.4\times$	+4638%
AE	FGSMR-AT	231.85	2012	$41.0\times$	+3996%
AE	PGD-AT	364.11	3167	$64.3\times$	+6333%
XGBoost	AdvRetraining	0.39	36.7	$1.03\times$	+3%
XGBoost	Feature Squeezing	—	—	Marginal	Marginal

[a]Feature Squeezing overhead was negligible and is reported qualitatively.

IV. CONCLUSIONS

This paper presented a five-stage evaluation pipeline for MLP, XGBoost, and AE classifiers in 5G network slices, integrating feature selection, statistically validated hyperparameter optimization, adversarial evaluation, and energy-aware defense analysis. Feature selection reduced the input dimensionality by more than 50% without compromising accuracy, while the Friedman and Nemenyi tests showed XGBoost significantly outperforming MLP and AE under clean conditions. All three models were ultimately driven to near chance-level accuracy, but at very different costs to the attacker, reflecting both architecture and attacker access (white-box PGD versus black-box Boundary Attack): breaking XGBoost demanded orders of magnitude more queries and energy than the gradient-based attacks sufficient against MLP and AE. Adversarial training substantially recovered the robustness of the neural models at considerable computational cost, whereas XGBoost achieved only partial recovery via AdvRetraining at a marginal 3% retraining-energy increase, on top of the offline cost of generating the reused adversarial examples.

Overall, model selection for 5G network slicing should jointly consider predictive performance, adversarial robustness, and computational efficiency: XGBoost is attractive for latency- and resource-constrained scenarios such as URLLC and mMTC, whereas adversarially trained neural models remain viable for centralized eMBB deployments with more computational resources.

The study is limited to the 5G-SliciNDD dataset (14,456 records); this modest size and the near-perfect baseline AUC (Table I) suggest the task may be easy, so findings should be validated on larger, more heterogeneous traffic. Future work includes additional 5G datasets, lightweight edge defenses, federated learning, and transfer-based, score-based, and generative attacks.

REFERENCES

[1] C.-C. Tseng, A. Mihovska, and S.-Y. Lien (Eds.), *The Road to B5G/6G Mobile Communication Networks: Technologies and Applications*. River Publishers, 2025.

[2] R. F. Olimid and G. Nencioni, "5G Network Slicing: A Security Overview," *IEEE Access*, vol. 8, pp. 99999–100009, 2020. doi: 10.1109/ACCESS.2020.2997702.

[3] M. Ahmadinejad, T. Azmin, and N. Shahriar, "5G Network Slice Type Classification using Traditional and Incremental Learning," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Miami, FL, USA, 2023, pp. 1–5. doi: 10.1109/NOMS56928.2023.10154385.

[4] Z.-X. Wu, Y.-Z. You, C.-C. Liu, and L.-D. Chou, "Machine Learning Based 5G Network Slicing Management and Classification," in *Proc. 2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Osaka, Japan, 2024, pp. 371–375. doi: 10.1109/ICAIIC60209.2024.10463325.

[5] J. Dias, P. Pinto, R. Santos, and S. Malta, "5G Network Slicing: Security Challenges, Attack Vectors, and Mitigation Approaches," *Sensors*, vol. 25, no. 13, p. 3940, 2025. doi: 10.3390/s25133940.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *Proc. International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[7] P.-S. Dang, P. T. Duy, and V.-H. Pham, "Investigating the Robustness Against Transferable Adversarial Attacks of Learning-Based Network Intrusion Detection System," in *Proc. 2024 RIVF International Conference on Computing and Communication Technologies (RIVF)*, 2024. doi: 10.1109/RIVF64335.2024.11009074.

[8] M. Parikh *et al.*, "Adversarial Vulnerabilities in Machine Learning: Differential Analysis of Gradient and XGBoost-Based Attack Mechanisms," in *Proc. 2025 International Conference on Intelligent Computing, Information and Control Systems (ICOIICS)*, 2025. doi: 10.1109/ICOIICS67115.2025.11390487.

[9] M. Ammar *et al.*, "5G-SliciNdd Dataset," *Figshare*, 2023. doi: 10.6084/m9.figshare.24446515.

[10] Scikit-learn Developers: PCA, scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, last accessed 2026/01/27

[11] Y. Alslman, S. H. Alnabelsi, and A. F. Otoom, "Breaking and Healing: GAN-Based Adversarial Attacks and Post-Adversarial Recovery for 5G IDSs," *IEEE Access*, vol. 13, pp. 132109–132127, 2025. doi: 10.1109/ACCESS.2025.3526414.

[12] L. Bouza, A. Bugeau, and L. Lannelongue, "How to estimate carbon footprint when training deep learning models? A guide and review," *Environmental Research Communications*, vol. 5, no. 11,p. 115014, Nov. 2023. https://doi.org/10.1088/2515-7620/acf81b

[13] Advanced Micro Devices, *AMD Ryzen 5 5500X3D*, 2025. https://www.amd.com/en/products/processors/desktops/ryzen/5000-series/amd-ryzen-5-5500x3d.html, Default TDP: 105 W, accessed 2026/06/14

[14] Y. Shi, Y. E. Sagduyu, T. Erpek, and M. C. Gursoy, "How to Attack and Defend NextG Radio Access Network Slicing With Reinforcement Learning," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 181–192, 2023.

[15] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[16] I. T. Jolliffe and J. Cadima, *Principal Component Analysis*, 2nd ed. Springer, 2016.

[17] W. Brendel, J. Rauber, and M. Bethge, "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[18] Trusted-AI, *Adversarial Robustness Toolbox (ART)*, GitHub repository. https://github.com/Trusted-AI/adversarial-robustness-toolbox, last accessed 2026/10/09