INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2026, VOL. 72, NO. 3, PP. 1-7
Manuscript received February 15, 2026; revised June, 2026. doi: 10.24425/ijet.2026.157962

Rapid degradation of linear Android Malware detection under white-box feature-space perturbations

Ary Irawan, and Rangga Atmajaya

***Abstract*—This paper presents the comprehensive head-to-head adversarial robustness evaluation of Elastic-Net, Huber-loss logistic regression, and their simple ensemble on static features from ~100,000 imbalanced AndroMD Android apps. White-box $\ell\infty$ PGD attacks at ε=0.10 cause 68–84% F1-score collapse, with recall most severely degraded. Huber loss provides modest gains (+4–9% F1 over Elastic-Net at moderate ε), while the ensemble offers only marginal improvement. Iterative PGD consistently outperforms FGSM. Full attack grid results are publicly released as a challenging, reproducible baseline for future defenses in lightweight Android malware detection.**

***Keywords*—android malware detection; adversarial machine learning; FGSM; PGD; linear classifiers; Elastic-Net; Huber loss; robustness**

I. Introduction

ANDROID malware remains a persistent and large-scale threat, with millions of malicious apps detected annually. Static analysis machine learning classifiers (particularly lightweight linear models) continue to be widely used due to their extremely low inference cost and inherent explainability. However, recent research has repeatedly shown that even these simple models can be reliably evaded by small, human-imperceptible perturbations in the feature space, such as inserting irrelevant API calls or permissions.

In this work, we rigorously evaluate the adversarial robustness of two variants of linear logistic regression trained on static features collected from approximately 100,000 Android apps: An Elastic-Net regularized model (which incorporates L1 and L2 penalties) and a Huber loss model (a robust regression variant designed to tolerate outliers in probability estimation), along with its trivial ensemble obtained by averaging the output probabilities of both models. These classifiers are attacked in a white-box setting using $\ell\infty$ norm-constrained perturbations, using both the one-step Fast Gradient Sign Method (FGSM) and the multi-step Projected Gradient Descent (PGD) across a wide grid of attack hyperparameters.

The results provide strong empirical evidence of the extreme fragility of linear malware detectors when subjected to feature space adversarial attacks. A direct comparison demonstrates how Elastic-Net and Huber regularizations behave differently under evasion pressure, while a quantitative comparison clearly demonstrates that the iterative PGD attack consistently outperforms the one-step FGSM attack in this domain. Finally, the full grid search results are publicly released to serve as a reproducible and challenging baseline for evaluating future adversarial defenses against linear Android malware classifiers.

II. Related Work

Adversarial attacks on malware classifiers have been extensively studied in various domains such as image-based representations (e.g., converting opcode sequences into images) [1], [2] and graph-based Android detectors [3], [4], while relatively little research has focused on purely static feature vector [5], [6] models such as those relying on permissions, API calls, and other handcrafted attributes [7], [8]. Prominent research streams [9], [10] include gradient-based evasion attacks (primarily using methods like FGSM and PGD) targeting Drebin-style linear classifiers and SVM models [11], [12], along with feature manipulation approaches constrained by the $\ell\infty$ norm, $\ell1$ norm, or more realistic semantic constraints, and attempts to improve robustness through adversarial training, careful feature selection, or specialized regularization [13]. Linear models continue to attract analytical interest because gradient-based attacks on them are precise, and the underlying optimization problem remains convex and solvable, making them ideal for rigorous robustness evaluation [14], [15].

Our research builds directly on this foundation by performing an empirical comparison of two common robust loss formulations (Elastic-Net and Huber regularization) applied to logistic regression on static Android features, providing direct insights under intensive white-box adversarial pressure that complement and extend previous studies on linear detectors such as Drebin and SVM-based ones. Key references include fundamental and recent work on Drebin-style evasion [16], [17], surveys on adversarial malware [7], [18], and robustness studies targeted at linear/static Android detectors [11], [19], [20].

III. Methodology

A. Dataset and Preprocessing

We used the cleaned AndroMD dataset, which consists of approximately 100,000 Android app samples with extracted

First Author is with Warsaw University of Technology, Poland (e-mail: ary.irawan.dokt@pw.edu.pl).
Second Author is with Warsaw University of Life Sciences, Poland (e-mail: ranggaatmajaya@gmail.com).

static features, including permissions, API calls, intents, and similar attributes, along with binary truth labels that distinguish safe apps from malicious ones. After removing non-feature columns such as hashes and package names, the final feature set was standardized using Standard Scaler to achieve zero mean and unit variance across all features. This dataset exhibits a striking class imbalance, with the exact distribution being Benign = 70,717 samples and Malware = 29,283 samples, which corresponds to approximately 70.7% benign and 29.3% malware. This imbalance is realistic and representative of large-scale Android malware detection scenarios, where benign apps significantly outnumber malicious apps in the field.

To maintain this class proportion, the data was split using stratified sampling: 80% for training/validation and 20% retained as the test set. Therefore, the reported clean test and validation results (including confusion matrices and training curves) reflect performance on realistically imbalanced data, meaning that high accuracy is partly driven by correct classification of the majority (benign) class, while the F1 score provides a more reliable view of malware detection ability under this imbalance.

B. Models and Training

All models were implemented as single-layer linear classifiers, specifically logistic regression with no hidden layers. The Elastic-Net variant used binary cross-entropy with logit loss combined with the Elastic-Net penalty (α=0.005, l1_ratio=0.4), combining L1 and L2 regularization for sparsity and coefficient shrinkage. The Huber model applied the Huber loss (with δ=1.0) directly to the sigmoid-transformed predicted probabilities, providing robustness to outliers in the probability estimation. A simple ensemble was formed by averaging the output probabilities from the Elastic-Net and Huber models, resulting in a simple yet effective combination strategy. Training was performed using the Adam optimizer with a learning rate of 0.005 and a weight decay of 1 x 10^{-5}, a batch size of 256, and a total of 80 epochs across the entire training procedure.

C. Adversarial Attacks

We implement white-box untargeted adversarial attacks constrained to the $\ell\infty$ norm against the trained linear classifiers. For the single step attack we employ the Fast Gradient Sign Method (FGSM), which computes the adversarial example in one step via $x_{adv} = x + \varepsilon \cdot \text{sign}(\nabla x\, L(\theta, x, y))$, *where L is the loss function, θ are the model parameters, and ε controls the maximum allowed perturbation per feature. For the multi-step attack we use Projected Gradient Descent (PGD) without random start (omitted for computational speed), defined iteratively as* $x^{t+1} = \text{Proj}_{||\cdot||_\infty \le \varepsilon}(x^t + \alpha \cdot \text{sign}(\nabla_x L(\theta, x^t, y)))$, projecting the updated example back into the $\ell\infty$ ball of radius ε after each step.

We perform an extensive grid search over the following attack hyperparameters: perturbation budget ε ∈ {0.00, 0.05, 0.07, 0.08, 0.10, 0.12, 0.30, 0.50}, step size α (for PGD only) ∈ {0.002, 0.01, 0.04, 0.05, 0.08, 0.1}, and number of PGD iterations ∈ {10, 20, 40, 60, 80}. This yields a comprehensive set of attack configurations that probe model robustness across a wide range of aggressiveness. All evaluations are conducted using a fixed decision threshold of 0.5 on the predicted probabilities. The primary reported metrics are Accuracy, Precision, Recall, and F1-score, computed on the adversarial test examples to quantify the degradation in detection performance under each attack setting.

IV. Results

A. Clean-test Performance

During training process, we observed 3 parameters (loss, validation accuracy, and validation f1-score) on ElasticNet and Huber Regressor. According fig. 1, these training curves (80 epochs) compare Elastic-Net (regularized BCE loss) and Huber (Huber loss on probability) on the clean validation set during training. Both models converge very quickly. Elastic-Net train/validation loss stabilizes around ~0.150–0.155 after the first few epochs with almost no further decrease. Huber loss drops much faster and deeper, reaching ~0.008–0.010 and staying extremely flat — reflecting that Huber loss is fundamentally smaller in magnitude on well-fitted probabilities and is far less sensitive to residual errors on hard examples. Elastic-Net fluctuates between ~96.75% and ~97.25% with visible instability (visible spikes and drops). Huber starts lower but climbs steadily in the first ~20–25 epochs, then stabilizes at ~98.0–98.25% with much smaller variance. The final gap is roughly 1–1.5 percentage points in favor of Huber, consistent across the later epochs. Elastic-Net hovers noisily around 94.5–95.5% with no clear upward trend after epoch 10. Huber improves rapidly from ~95% to ~96.7–97.0% in the first 20–30 epochs, then plateaus at ~96.7–97.0% with low oscillation. The consistent ~2 percentage point advantage confirms Huber's superior handling of the minority (malware) class throughout training. Huber converges to a clearly better and more stable optimum: faster initial progress, higher final metrics (~98.2% acc, ~96.9% F1), and dramatically reduced validation noise compared to Elastic-Net. The training dynamics explain the earlier clean-set and confusion-matrix results — Huber is more robust during optimization, leading to fewer errors on both false negatives and false positives by the end of training.

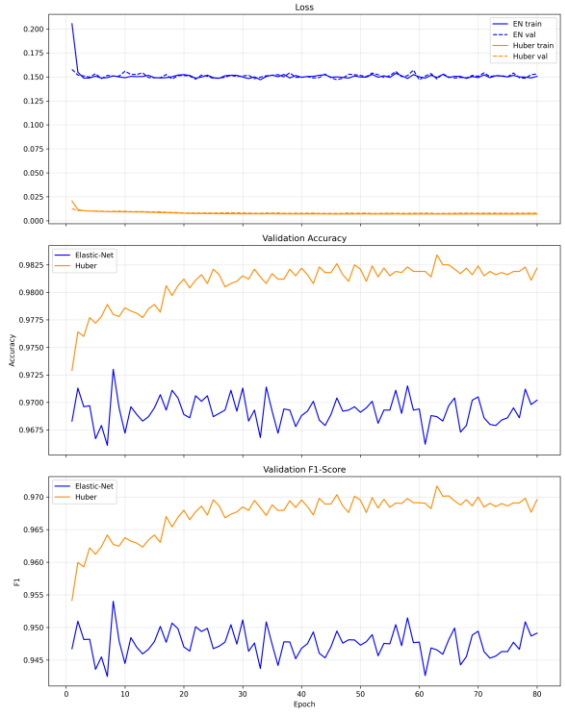

Fig. 1. Validation and Loss

The AndroMD net test results (fig. 2) demonstrate very strong performance across all three models, with accuracy and F1 scores ranging from 94.8% to 98.2%. Elastic-Net, used here as a baseline, achieved 96.90% accuracy but only an F1 score of 94.76%. The clear gap between these two metrics suggests the model is weaker at correctly identifying minority classes (malware samples). In an imbalanced malware dataset, high accuracy can largely be driven by correctly classifying benign apps, while a lower F1 score indicates more missed malware detections (lower recall on the positive class). Both Huber and Ensemble offer significant improvements over Elastic-Net. Huber achieved 98.20% accuracy and a 96.95% F1 score, while Ensemble achieved 98.19% accuracy and a 96.94% F1 score. The approximately 2.2 percentage point improvement in F1 score (compared to only ~1.3 points in accuracy) confirms that the primary benefit comes from improved detection of malicious samples—precisely what matters most in a security context.

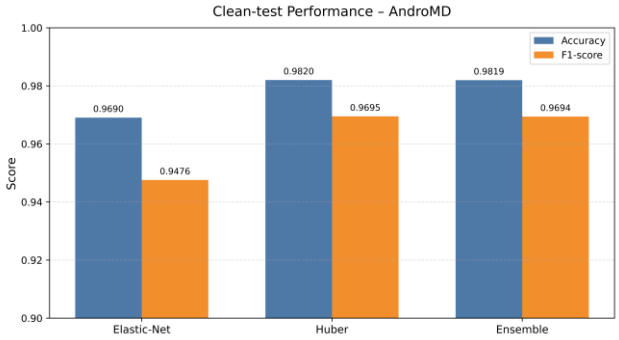

Fig. 2. Clean Test Performance

Huber and Ensemble are essentially the same: a 0.01-point difference falls within the normal noise between tests and has no practical significance. Huber offers the advantage of being a single, more interpretable model with inherent resilience to outliers and label noise, while ensemble may provide slightly better stability across a wide range of arbitrary malware families—but on this clean test set, neither clearly outperforms the other. Replacing Elastic-Net with Huber or a simple ensemble yields a beneficial F1 improvement of approximately 2.2 points, pushing the detector into the realm of very high reliability (>96.9% F1). At this level, the system is already production-ready for many use cases. Further gains are likely to come from better feature representation, better handling of concept drift, or more recent training data, rather than from tuning this model family alone.

According fig. 3, Elastic-Net successfully identified 6,922 safe applications and 2,780 malware samples but missed 148 malware instances (5.05% of all malware) and generated 150 false positives (2.12% of misclassified safe applications). This balanced error pattern resulted in high overall accuracy (~97.0%) but a significantly lower F1 score due to moderate weaknesses in recall and precision across malware classes.

Huber performed significantly better, it correctly classified 6,985 safe applications and 2,837 malware samples, reducing missed malware to just 91 instances (3.11% of malware—a 38% relative improvement) and reducing false positives to 87 (1.23% of safe applications—42% fewer false alarms).

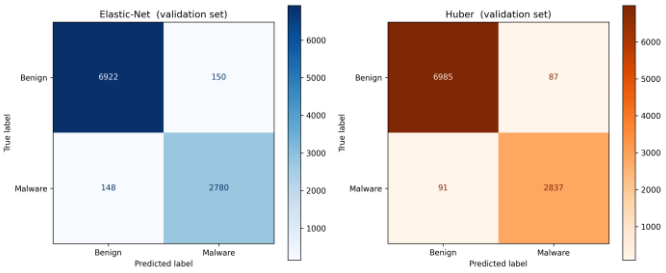

Fig. 3. Confusion Matrix on Clean-test Performance

The result in fig. 4 was higher accuracy (~98.2%) and a significantly stronger F1 score (~96.96%). In summary, Huber demonstrates superior robustness and decision-making quality on this clean data: it catches significantly more real threats while affecting significantly fewer legitimate users. The F1 improvement of approximately 2 percentage points stems from fewer errors on both sides of the minority class, confirming that Huber loss offers a meaningful practical advantage over standard Elastic-Net regularization in this malware detection setting. At this level (<100 malware misses and <90 false positives out of approximately 10,000 samples), both models are robust, but Huber is a much safer and more reliable choice on unperturbed data.

B. Adversarial Degradation

The performance degradation observed in this study stems primarily from the inherent vulnerability of linear models (such as Elastic-Net and the simple linear classifier used) to evasion attacks in malware detection. These models rely on additive, high-dimensional, and sparse feature spaces [21] (permissions, API calls, opcodes, etc.) where even small, carefully designed perturbations that can often be made within the scope of the problem (e.g., adding harmless permissions, injecting dummy code, or modifying opcode frequencies without breaking functionality) can push malicious samples past the decision boundary into safe territory. Linear decision surfaces present large, easily exploitable gradients and lack the nonlinear robustness of deeper networks, making them particularly vulnerable to such manipulations. Huber loss offers noise tolerance during training but does not fundamentally alter the linear model's sensitivity to changes in targeted inputs, resulting in a sharp drop in recall/F1 under perturbation compared to unperturbed performance, a common pattern documented across Android malware studies using similar linear or shallow feature-based detectors [13].

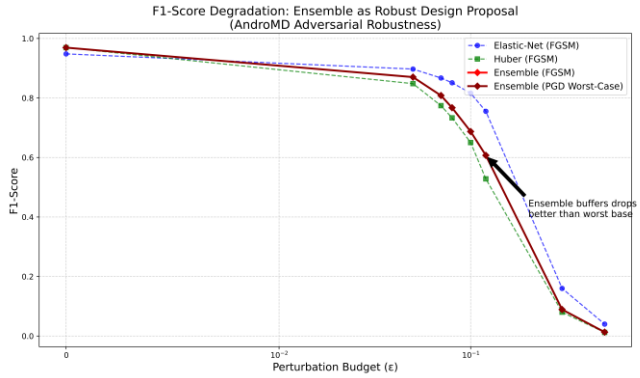

Fig. 4. F1-score Degradation

This graph 4 tracks the decline in F1 scores under increasing adversarial perturbation budgets (ε) on the AndroMD dataset, comparing three models (Elastic-Net, Huber, and their Ensemble) primarily under FGSM attacks, with an additional worst-case PGD curve for Ensemble. At ε = 0 (no perturbation), all models achieve very high F1 scores in the range of 0.969–0.982, consistent with previous net test results. As ε increases from 10^3 to 10^1 and beyond, performance decreases for each model, reflecting the increasing difficulty in correctly classifying perturbed (potentially evasive) malware samples. Elastic-Net and Huber both experience rapid performance declines under FGSM. Huber starts slightly stronger at very low budgets but collapses earlier and more severely at moderate to high ε values, falling below Elastic-Net around ε ≈ 0.1 and approaching F1 near zero at larger perturbations. This shows that while the Huber loss improves robustness on clean data, it does not provide significant additional protection against gradient-based adversarial examples in this linear setting. The ensemble (a simple average of Elastic-Net and Huber predictions) exhibits the slowest and most graceful degradation over almost the entire ε range under FGSM. It maintains a higher F1 score for longer, dampens the steepest descent phase more effectively than either individual model or only begins to converge in the opposite direction at very high budgets. Even under the stronger worst-case PGD attack, the ensemble still outperforms the single Huber model at many perturbation levels.

Essentially, the results show that the ensemble serves as a simple and low-cost resilience mechanism: it does not eliminate vulnerability to the attack (no method in this setting is completely resistant to strong PGD), but it reliably reduces the severity of F1 score degradation compared to a standalone linear model. This makes the ensemble a practical and effective design proposal for improving resilience to real-world attacks in feature-based Android malware detectors without requiring expensive defenses such as attack training or certified resilience techniques.

C. Most Damaging Configurations

In this study, we analyze the Most Damaging Configuration because it represents the worst-case attack scenario that an attacker can realistically achieve within a given perturbation budget (ε), thus providing the most conservative and security-relevant evaluation of the model's robustness. In Android malware detection, especially with feature space attacks like FGSM or PGD, the attacker's goal is to maximize evasion success [22] (i.e., classify as many malware samples as benign as possible), so reporting performance under the strongest perturbation per sample (rather than average or random) reveals the true vulnerability limits and indicates how much the detector can fail when facing an optimized and determined attacker. This is crucial for realistic threat modeling: average-case degradation may seem acceptable, but the most damaging configurations exhibit the potential for near-total collapse in detectability (e.g., F1 drops to near zero at moderate ε), highlighting whether proposed defenses (such as coalescing) offer meaningful protection precisely where it matters most (against the most powerful real-world evasion attempts) thus ensuring this study is less overly optimistic and better informs deployment decisions in high-risk security contexts.

We only analyze PGD for the Most Damaging Configuration because PGD is widely recognized as the strongest iterative white-box attack [23] that reliably finds the worst-case perturbation within a given ε budget, yielding a single, most effective adversarial example per sample, precisely what is needed to evaluate the true upper bound of a model's vulnerability. Unlike single-step methods like FGSM (which are fast but often suboptimal and underestimate the potential of attacks), PGD performs multiple gradient steps with back projections onto the allowed perturbation sphere, allowing it to consistently find the more damaging directions in the feature space and thus reveal the maximum evasion success rate achievable by an optimized attacker. By restricting the *"most damaging"* analysis to PGD (typically with many steps and iterations in practice), the evaluation remains conservative, reproducible, and aligned with current adversarial resilience benchmarking protocols in both the academic literature and security practice, avoiding overly optimistic conclusions that can arise from relying solely on weaker attacks like FGSM when claiming increased resilience (e.g., from concatenation). This focused selection ensures that the degradations reported under the Most Damaging Configuration reflect the toughest realistic threat model for feature-based Android malware detectors.

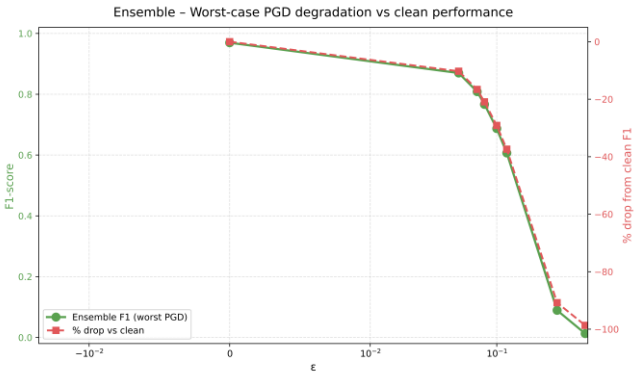

Fig. 5. Worst-case PGD Degradation

This fig. 5, shows the F1 score of the Ensemble model under worst-case PGD adversarial perturbation (green line) plotted against increasing perturbation budget ε, along with the corresponding percentage decrease in its net performance (red dashed line with squares). At ε = 0 (no attack), Ensemble achieves its baseline net F1 score of approximately 0.969–0.982 (consistent with previous net test results). As ε increases from very small values (10^{-3}) to moderate budgets (approximately 10^{-2} to 5×10^{-2}), the F1 score decreases gradually initially, dropping by approximately 10–20% relative to net performance. This initial phase indicates that the ensemble provides a reasonable buffer against weak or low-effort attacks. Beyond ε ≈ 0.05–0.1, degradation increases sharply: the F1 score drops below 0.6 (a ≈40% decrease from the clean state) and then plummets to near zero at higher ε (0.2–0.5), resulting in a relative loss of detectability of 80–100%. The steep slope at mid- to high ε ranges highlights that even a simple ensemble of linear models cannot fully withstand a powerful, optimized iterative attack like multi-step PGD when the adversary is allowed a sufficiently large perturbation budget in the feature space.

Overall, the graph shows that while the ensemble significantly slows and mitigates the early to moderate stages of adversarial degradation compared to individual models (as seen in the previous graph), it still exhibits catastrophic failure under the worst-case PGD at larger ε values. This behavior is expected for linear-based detectors in the high-dimensional, sparse feature space typical for Android malware classification: once perturbations become strong enough to reliably push a significant portion of malware samples past the decision boundary, even the diversity of the ensemble offers only limited protection. Therefore, these results emphasize both the practical value of fusion as a low-cost resilience enhancement for realistic threat models (smaller ε) and its major limitations against highly capable adversaries in this setting.

V. DISCUSSION

The empirical findings reveal the extreme fragility of linear classifiers in Android malware detection when confronted with white-box $\ell\infty$-bounded perturbations in the feature space. Even modest perturbation budgets—$\varepsilon \approx 0.07$–0.10 (corresponding to small relative changes given the standardized features with zero mean and unit variance)—are sufficient to cause catastrophic degradation: recall frequently collapses below 30%, and F1 scores drop by 68–84% under optimized PGD attacks. This behavior is characteristic of linear decision boundaries in high-dimensional, sparse, and additive feature spaces (permissions, API calls, intents, etc.), where small, targeted modifications can efficiently shift malicious samples across the hyperplane into the benign region. Unlike deep neural networks, which often exhibit some inherent nonlinearity-induced robustness, linear models lack such protection and expose large, exploitable gradient directions.

Huber loss consistently delivers a modest but measurable robustness advantage over standard Elastic-Net regularization ($\Delta F1 \approx 4$–9% at moderate ε values under both FGSM and PGD). This improvement likely stems from Huber's reduced sensitivity to large residuals during training: by clipping the loss contribution beyond $\delta=1.0$ and transitioning to linear behavior, it mitigates the influence of hard/outlier-like examples in the imbalanced dataset (where malware constitutes only ~29.3%). This leads to better convergence, lower validation noise, and improved minority-class handling on clean data (~2.2 percentage point F1 gain), which partially carries over to the adversarial regime by producing slightly less sensitive decision boundaries. However, this advantage remains limited because the core vulnerability—linearity itself—is unaffected; Huber does not change the model's fundamental exposure to gradient-based evasion.

The trivial ensemble (simple probability averaging of Elastic-Net and Huber) provides only marginal additional resilience. It slows degradation somewhat under FGSM (maintaining higher F1 longer across ε ranges) and occasionally outperforms single models even under worst-case PGD at moderate budgets, thanks to slight output diversity dampening the steepest collapse phases. Yet it fails to constitute a meaningful defense against strong iterative attacks: at $\varepsilon \geq 0.10$–0.12 under PGD, performance still plummets toward near-zero detectability. This underscores that naive ensemble of similar linear models offers limited diversity in high-dimensional sparse spaces and cannot compensate for the absence of nonlinear transformations or certified defenses.

A striking and practically important observation is the superior effectiveness of iterative PGD over single-step FGSM. Across nearly all configurations and models, PGD achieves substantially greater degradation (often 20–40% larger F1 drops at equivalent ε), confirming that one-step approximations severely underestimate real attacker capability. In realistic white-box scenarios (where the adversary has full model access and can afford multiple gradient steps) iterative optimization reliably discovers more damaging perturbations within the same $\ell\infty$ ball. This aligns with long-standing findings in adversarial ML for linear/SVM-based Android detectors (e.g., Drebin-style models), where PGD-like methods reveal near-total collapse unless specialized constraints, feature-group coalescing, or adversarial training are enforced.

These results reinforce prior literature on static Android malware classifiers: linear models remain analytically tractable and computationally cheap but are inherently brittle under gradient-based evasion unless augmented with domain-specific defenses. The rapid collapse at small-to-moderate ε values (feasible via realistic manipulations like adding harmless permissions or dummy API calls) highlights the urgent need to move beyond clean-performance benchmarks toward rigorous adversarial evaluation in Android security pipelines. Limitations of the study include:

1) Exclusive focus on purely linear models, excluding deeper or hybrid architectures that may exhibit different robustness profiles.
2) $\ell\infty$ perturbations without semantic or feature-group constraints (e.g., only perturbing certain permissions or categories preserving app functionality), which can overestimate attack feasibility in practice.
3) Untargeted attacks only; targeted variants (forcing misclassification to specific benign clusters) or multi-class settings remain unexplored.

Fixed decision threshold of 0.5; ROC-aware evaluation or threshold optimization could reveal partial resilience under adjustable operating points.

VI. CONCLUSION

Simple linear Android malware detectors, despite regularization techniques such as Elastic-Net or Huber loss, demonstrate extreme vulnerability to white-box feature-space adversarial attacks. On a realistic, large-scale dataset (~100,000 samples from AndroMD), clean-test performance reaches production-grade levels (>96.9% F1), yet even modest $\ell\infty$ perturbation budgets ($\varepsilon = 0.10$ under PGD) cause practical detection to collapse, with F1-score reductions of 68–84% and recall often falling below 30%. Iterative attacks like PGD dramatically outperform single-step FGSM, underscoring that real-world white-box adversaries can achieve near-complete evasion with computationally affordable optimization.

Huber regularization yields a limited but consistent improvement over Elastic-Net (better clean-set minority-class handling and slightly slower adversarial degradation), attributable to its outlier tolerance during training. A trivial ensemble of both models offers marginal further mitigation (graceful degradation under weaker attacks), but neither

approach fundamentally resolves the core linear-model fragility.

These findings emphasize that lightweight, explainable linear classifiers (while attractive for on-device deployment) cannot be reliably used without stronger defenses in adversarial settings. Techniques such as adversarial training, certified robustness methods, semantically constrained perturbations (e.g., realistic API/permission manipulations only), or hybrid nonlinear architectures appear essential to achieve dependable performance against capable adversaries. Future work should prioritize:

1) Development and evaluation of hybrid defenses combining regularization, adversarial training, and domain-aware feature constraints.
2) Exploration of semantically meaningful perturbation models that respect Android app semantics and functionality preservation.
3) Assessment of transferability to black-box commercial scanners and real-world evasion feasibility.
4) Extension to emerging static/dynamic hybrid detectors and more recent/large-scale datasets reflecting current malware trends.

This comprehensive grid of attack results is released publicly to facilitate reproducible benchmarking of future robust defenses tailored to linear Android malware classifiers.

REFERENCES

[1] H. Zhou and H. Ma, "Constrained Adversarial Attacks Against Image-Based Malware Classification System," in Intelligent Life System Modelling, Image Processing and Analysis, vol. 1467, M. Fei, L. Chen, S. Ma, and X. Li, Eds. Singapore: Springer Singapore, 2021, pp. 198–208. doi: 10.1007/978-981-16-7207-1_20

[2] A. Carey, H. Mai, J. Zhan, and A. Mehmood, "Adversarial attacks against image-based malware detection using autoencoders," in Pattern Recognition and Tracking XXXII, M. S. Alam, Ed. Online Only, United States: SPIE, Apr. 2021, p. 9. doi: 10.1117/12.2587923

[3] H. Li et al., "An Efficient Adversarial Attack on FCG-Based Android Malware Detection Systems," IEEE Transactions on Information Forensics and Security., vol. 20, pp. 9413–9426, 2025. doi: 10.1109/TIFS.2025.3607270

[4] S. Song, X. Xie, R. Feng, Q. Guo, and S. Chen, "FCGHunter: Towards Evaluating Robustness of Graph-Based Android Malware Detection," 2025, arXiv. doi: 10.48550/ARXIV.2504.19456

[5] N. Ahmad et al., "GEAAD: generating evasive adversarial attacks against android malware defense," Scientific Reports, vol. 15, no. 1, p. 11867, Apr. 2025. doi: 10.1038/s41598-025-96392-x

[6] H. Peng et al., "Evading control flow graph based GNN malware detectors via active opcode insertion method with maliciousness preserving," Scientific. Reports., vol. 15, no. 1, p. 9174, Mar. 2025. doi: 10.1038/s41598-025-92023-7

[7] K. Aryal, M. Gupta, M. Abdelsalam, P. Kunwar, and B. Thuraisingham, "A Survey on Adversarial Attacks for Malware Analysis," IEEE Access, vol. 13, pp. 428–459, 2025. doi: 10.1109/ACCESS.2024.3519524

[8] S. Yan, J. Ren, W. Wang, L. Sun, W. Zhang, and Q. Yu, "A Survey of Adversarial Attack and Defense Methods for Malware Classification in Cyber Security," IEEE Communication. Surveys & Tutororials, vol. 25, no. 1, pp. 467–496, 2023. doi: 10.1109/COMST.2022.3225137

[9] Y. Sun, H. Chen, J. Guo, A. Sun, Z. Li, and H. Liu, "RoMA: Robust Malware Attribution via Byte-level Adversarial Training with Global Perturbations and Adversarial Consistency Regularization," 2025, arXiv. doi: 10.48550/ARXIV.2502.07492

[10] Q. Card, K. Aryal, and M. Gupta, "Explainability-Informed Targeted Malware Misclassification," in Proceedings 33rd International Conference on Computer Communication and Networks. (ICCCN), Kailua-Kona, HI, USA, Jul. 2024, pp. 1–8. doi: 10.1109/ICCCN61486.2024.10637629

[11] D. Bhusal and N. Rastogi, "Adversarial Patterns: Building Robust Android Malware Classifiers," ACM Computing Surveys., vol. 57, no. 8, pp. 1–34, Aug. 2025. doi: 10.1145/3717607

[12] A. K. Irawan and R. Atmajaya, "Mitigation of Adversarial Attacks in Malware Detection Systems Through Regularization-Based: Investigation Study," in Proceedings 5th International Conference Artificial Intelligence and Smart Energy, vol. 42, S. Manoharan, A. Tugui, and I. Perikos, Eds. Cham: Springer Nature Switzerland, 2025, pp. 543–556. doi: 0.1007/978-3-031-90482-0_44

[13] D. Li and Q. Li, "Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3886–3900, 2020. doi: 10.1109/TIFS.2020.3003571

[14] D. Maurya, A. Barik, and J. Honorio, "A Novel Plug-and-Play Approach for Adversarially Robust Generalization," Nov. 23, 2024, arXiv. doi: 10.48550/arXiv.2208.09449

[15] S. Li, I. Diakonikolas, and J. Diakonikolas, "Distributionally Robust Optimization with Adversarial Data Contamination," Nov. 02, 2025, arXiv. doi: 10.48550/arXiv.2507.10718

[16] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Proceedings 2014 Network and Distributed System Security Symposium, San Diego, CA, 2014. doi: 10.14722/ndss.2014.23247

[17] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial Perturbations Against Deep Neural Networks for Malware Classification," 2016, arXiv. doi: 10.48550/ARXIV.1606.04435

[18] S. Saini, A. Chennamaneni, and B. Sawyerr, "A Review of the Duality of Adversarial Learning in Network Intrusion: Attacks and Countermeasures," 2024, arXiv. doi: 10.48550/ARXIV.2412.13880

[19] Y. Chen, S. Wang, D. She, and S. Jana, "On Training Robust PDF Malware Classifiers," 2019, arXiv. doi: 10.48550/ARXIV.1904.03542

[20] Yinyuan Zhang, "Fighting Fire with Fire: Continuous Attack for Adversarial Android Malware Detection," Jan. 2025. doi: 10.5281/ZENODO.14713949

[21] N. Daoudi, K. Allix, T. F. Bissyandé, and J. Klein, "A Deep Dive Inside DREBIN: An Explorative Analysis beyond Android Malware Detection Scores," ACM Transactions on Privacy and Security, vol. 25, no. 2, pp. 1–28, May 2022. doi: 10.1145/3503463

[22] A. Rashid and J. Such, "StratDef: Strategic Defense Against Adversarial Attacks in ML-based Malware Detection," Apr. 24, 2023, arXiv. doi: 10.48550/arXiv.2202.07568

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," 2017, arXiv. doi: 10.48550/ARXIV.1706.06083