BRICS Journal of Applied Mathematics and Engineering Sciences
Volume 1 • Issue 1 • Year 2026
ISSN: XXXX-XXXX | DOI:

Lightweight Hybrid Side-Channel Anomaly Detection for Edge AI Accelerators: A Simulation-Based Machine Learning Study

Harshita Mandloi[1]
[1]Indipendent Researcher harshitajmandloi@gmail.com

ABSTRACT

As Artificial Intelligence (AI) increasingly operates on resource-limited accelerators, the analysis of physical leakage and the possibility of hardware-level tampering becomes increasingly relevant. This paper presents a proof-of-concept simulation-based study of a lightweight hybrid approach to detecting Trojan activation, voltage glitching, clock glitching, and memory-bus probing. A physically parameterized synthetic dataset of 12,000 traces was generated using a combination of compute bursts, workload variation, measurement noise, and attack-specific perturbations. Fourteen low-cost time- and frequency-domain features were extracted from 256-sample windows. Logistic regression, random forest, gradient boosting, and radial-basis-function support vector machines were evaluated in single-model and fusion settings, with particular emphasis on a weighted probability combination. Using a 75/25 train/test split, the final hybrid model achieved 96.57% accuracy, 98.01% precision, 95.07% recall, 96.51% F1-score, and 0.9939 AU-ROC on the test set. On normal traces, the model demonstrated 98.07% specificity, while the attack-wise true-positive rates were 93.60% for Trojan activation, 89.33% for voltage glitches, 97.33% for clock glitches, and 100% for bus probing. The results indicate that a reduced set of statistical and spectral properties of power traces can facilitate hardware-level integrity screening. Extended physical-layer analysis using FPGA, GPU, and microcontroller platforms is recommended for future research.

Keywords: Edge AI, Hardware Security, Side-Channel Analysis, Anomaly Detection, Power Traces, Machine Learning, AI Accelerators

1. INTRODUCTION

Edge AI has offloaded significant parts of neural-network computation from centralized servers to embedded processors, microcontrollers, GPUs, and other accelerators. While reducing latency and bandwidth requirements, such a trend makes neural networks susceptible to physical and microarchitectural side channels that are invisible at the software level. Recent works have demonstrated that electromagnetic, cache, memory, power, and timing side channels can disclose neural-network architecture, weights, or runtime behavior. Specifically, CSI NN showed that electromagnetic side channels could expose the structural information about neural networks [1]. DeepSniffer and Cache Telepathy demonstrated that neural-network architectural hints and shared resource behaviors could be decoded with high confidence, significantly reducing the search space for model extraction [2] [3]. Similarly, Leaky Nets showed that power and timing measurements could disclose embedded neural-network models and their inputs [4].

The same physical layer could be used to mount integrity attacks, for example, bit-flip attacks or Rowhammer-style hardware faults, which could corrupt a small percentage of model weights, significantly affecting the model's behavior [5] [6]. Memory-side-channel attacks such as DeepSteal highlighted that model confidentiality and integrity were closely intertwined, as an attacker could first extract some of the model's weights and then use that information to target the model integrity [7]. With that, we posit that a runtime integrity controller should be placed as close to the accelerator as possible to observe and react to physical-layer anomalies that could herald poisoned inferences.

Most recent works in this area either focus on attack demonstrations or require expensive data processing before anomalous behavior can be detected. For embedded platforms, such solutions may be inappropriate because of processing-power and memory limitations. A detector for embedded platforms should have a small footprint and remain effective under substantial benign workload variation. Accordingly, this work examines whether such a detector can be built using a low-dimensional representation of power traces. To evaluate this hypothesis, we employ a synthetic trace generator that captures realistic accelerator compute patterns and parameterized attack scenarios, enabling a focused comparison while limiting the resource cost and avoiding premature experimentation on real-world accelerators.

The main contributions of the study are:

- A reproducible simulation framework: 12,000 normalized power-like traces are generated for normal inference and four hardware-security event classes under workload jitter and measurement noise.
- A lightweight feature set: fourteen time-domain and spectral descriptors are computed from short windows, avoiding the memory and training cost of raw-trace deep neural networks.
- A hybrid detector: probability outputs from logistic regression, random forest, gradient boosting, and RBF-SVM models are fused to improve robustness across attack types.
- Attack-wise evaluation: overall metrics are complemented by class-specific detection rates, specificity analysis, ROC comparison, and feature-importance interpretation.

2. PROBLEM FORMULATION AND THREAT MODEL

2.1 Edge-AI Execution Model

Consider an edge accelerator running a fixed or slowly changing neural-network inference task. During normal operation, repeated matrix multiplication, convolution, activation, and memory-transfer phases produce a characteristic but non-identical power profile. Let x(t) be the normalized physical signal observed in an inference window. The baseline model used in this study represents the signal as a superposition of workload-dependent computation bursts, a low-frequency platform component, and measurement noise:

$$x(t) = P_0 + \Sigma_i\, a_i\, exp[-(t-\mu_i)^2/(2\sigma_i^2)] + s(t) + \varepsilon(t). \quad (1)$$

Here, P_0 is the baseline activity level, a_i, μ_i, and σ_i define the amplitude, center position, and width of computation bursts, s(t) encodes the lower frequency activity, and ε(t) is a noise term. For each simulated inference, amplitudes, positions, and burst widths were randomly varied in order to generate realistic intra-class diversity, rather than using one prototype waveform.

2.2 Security Events

Four abnormal conditions were modeled because they represent different physical phenomena. Trojan activation introduces a weak periodic component plus a local spike; voltage glitching produces a temporary decrease followed by recovery; clock glitching produces a local stretching of the temporal coordinate plus a short high-frequency spike; and bus probing generates transient high-frequency activity with a small regular shift superimposed on normal activity. These surrogates should be considered parameterized attack signatures rather than measured traces from a particular device. The detector was not trained to distinguish among attack types; instead, it answers a binary question: normal or abnormal. Attack-wise labels were retained only to evaluate class-specific sensitivity.

BRICS Journal of Applied Mathematics and Engineering Sciences
Volume 1 • Issue 1 • Year 2026
ISSN: XXXX-XXXX | DOI:

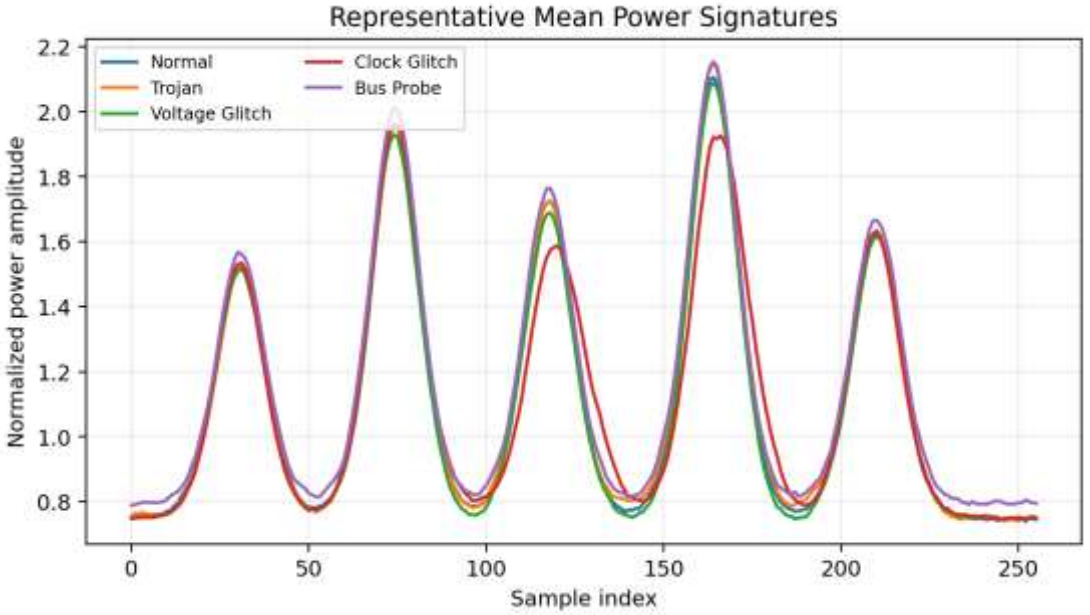

Fig. 1: Representative mean simulated power signatures for normal inference and four hardware-security event classes.

3. DATASET GENERATION AND FEATURE EXTRACTION

3.1 Synthetic Dataset Configuration

The dataset consists of 12,000 traces of 256 samples each. The normal class consists of 6,000 traces, and the remaining classes are divided equally among Trojan activation, voltage glitching, clock glitching, and bus-probing conditions, resulting in 1,500 traces for each attack type. A fixed random seed was used to obtain trace sets with identical statistical properties for reproducibility. Normal traces were generated with ±15% workload scaling, random burst-amplitude variation, small timing jitter, variable burst width, and Gaussian measurement noise. Finally, attack perturbations were added to a freshly generated normal trace in order to prevent the detector from using a separate baseline distribution for attacks.

Table 1: Simulation Dataset and Signal Configuration

Component	Configuration	Purpose	Count / Range
Trace length	256 samples	Short runtime monitoring window	256
Normal inference	Jittered compute bursts + noise	Baseline workload variability	6,000 traces
Trojan activation	Periodic leakage + local trigger pulse	Stealthy additional hardware activity	1,500 traces
Voltage glitch	Negative pulse + rebound	Transient supply manipulation	1,500 traces
Clock glitch	Local time warp + HF burst	Timing disturbance	1,500 traces
Bus probing	Gated HF oscillation + baseline shift	Memory/interface observation activity	1,500 traces

3.2 Lightweight Feature Vector

Each trace was characterized by a 14-dimensional feature vector. The time-domain features included mean, standard deviation (SD), root-mean-square value, peak factor, kurtosis, skewness, lag-1 autocorrelation, crest range, maximum first difference, first-difference SD, and second-difference or "jerk" SD. The frequency-domain features included normalized spectral entropy, high-frequency energy ratio, and mid-band energy ratio. Normalized spectral entropy was estimated from the normalized power spectrum p_k as:

$$H_s = -\Sigma_k\, p_k \ln(p_k) / \ln(K). \quad (2)$$

The high-frequency and mid-band ratios were extracted as proportions of spectral energy outside the expected compute-burst region. These spectral ratios are suitable for detecting periodic probing and clock manipulation, while derivative-based features emphasize transient faults. The feature set was designed to remain interpretable and low-cost: except for the 256-point real Fast Fourier Transform (FFT), all other descriptors involve only linear-time operations.

4. MACHINE LEARNING FRAMEWORK

4.1 Baseline Models

Four classifiers were selected, each reflecting a different decision mechanism. Logistic regression provides a linear probabilistic baseline. Random forest captures nonlinear interactions through an ensemble of decision trees. Gradient boosting sequentially fits weak learners to difficult cases. The radial-basis-function support vector machine provides a nonlinear margin-based classifier after feature standardization. The dataset was divided using a stratified 75:25 split, yielding 9,000 training traces and 3,000 independent test traces. None of the test traces were used for model fitting or probability fusion.

4.2 Proposed Weighted Hybrid Detector

The proposed detector fuses calibrated class probabilities from the four base models. Let pLR, pRF, pGB, and pSVM denote predicted attack probabilities. The final anomaly score is:

$$S = 0.20\, pLR + 0.10\, pRF + 0.10\, pGB + 0.60\, pSVM. \quad (3)$$

A trace is marked as anomalous when S is at or above 0.5. The RBF-SVM achieved the best single-model discrimination during model development and therefore received the largest fusion weight. The remaining learners primarily add diversity in borderline cases. The weighting scheme was kept simple and deterministic, allowing the detector to be reproduced without a secondary meta-learner.

BRICS Journal of Applied Mathematics and Engineering Sciences
Volume 1 • Issue 1 • Year 2026
ISSN: XXXX-XXXX | DOI:

4.3 Evaluation Measures

Evaluation included accuracy, precision, recall, F1-score, specificity, and area under the receiver operating characteristic curve (AUC). In security monitoring, precision is the proportion of raised alerts that correspond to true attacks, while recall is the proportion of actual attacks correctly detected. Specificity is the proportion of normal instances correctly identified. Attack-wise detection rates were also examined for each of the four attack classes to reveal failure patterns that can be obscured by aggregate performance metrics.

5. RESULTS AND DISCUSSION

5.1 Comparative Detection Performance

All four base classifiers separated normal and abnormal instances with high accuracy, although their precision-recall characteristics differed. Logistic regression achieved 96.17% accuracy with an AUC of 0.9932, indicating strong discrimination across decision thresholds. Random forest and gradient boosting produced slightly lower accuracy than logistic regression and the radial-basis-function support vector machine (RBF-SVM), while maintaining high precision and therefore relatively conservative alert behavior. RBF-SVM provided the strongest single-model performance, with accuracy, precision, recall, and F1-score of 96.53%, 98.34%, 94.67%, and 96.47%, respectively.

Table 2: Test-Set Performance of Baseline Models and Proposed Hybrid Detector

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	AUC	Prediction (µs/trace)*
Logistic regression	96.17	97.79	94.47	96.10	0.9932	0.23
Random forest	94.60	97.38	91.67	94.44	0.9869	24.77
Gradient boosting	94.83	97.33	92.20	94.69	0.9888	2.80
RBF-SVM	96.53	98.34	94.67	96.47	0.9933	33.85
Proposed hybrid	96.57	98.01	95.07	96.51	0.9939	—

**Prediction time is a Python desktop baseline and is reported only for relative comparison; it is not an edge-device latency claim.*

Probability fusion improved on the best-performing single classifier by a small but consistent margin. The proposed hybrid achieved 96.57% accuracy and increased recall to 95.07% while maintaining 98.01% precision. An AUC of 0.9939 indicates that the fused score retains strong discriminative power across decision thresholds. Although the gain over RBF-SVM is modest, fusion reduces dependence on the decision geometry of a single learning algorithm and may improve robustness when trace distributions shift across workloads.

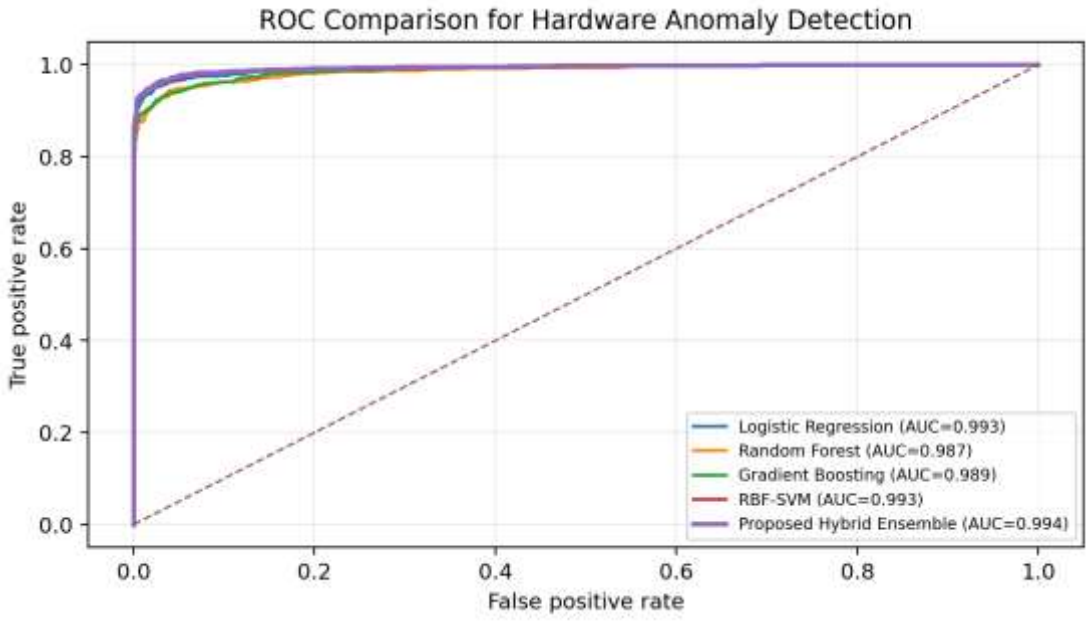

Fig. 2: ROC comparison of the four baseline classifiers and the proposed weighted hybrid detector on the independent test set.

5.2 Attack-Wise Sensitivity

The confusion matrix of the proposed detector contained 1,471 true-normal decisions, 29 false alarms, 1,426 correctly detected attack traces, and 74 missed attacks, corresponding to 98.07% specificity. Attack-wise analysis showed that bus-probing signatures were the easiest to identify because their gated high-frequency activity strongly modifies spectral energy. Clock glitches were also detected reliably because local time warping and high-frequency disturbances affect derivative and spectral descriptors. Trojan activation was more subtle but achieved a 93.60% detection rate. Voltage glitches had the lowest detection rate at 89.33%, because short disturbances can occur between major compute bursts and may resemble ordinary noise or burst-timing variability.

Table 3: Class-Specific Detection Behaviour of the Proposed Hybrid Model

Class	Test traces	Correct decision (%)	Interpretation
Normal	1,500	98.07 specificity	Low false-alarm rate
Trojan activation	375	93.60 detection	Subtle periodic leakage is detectable
Voltage glitch	375	89.33 detection	Most difficult transient condition
Clock glitch	375	97.33 detection	Strong timing and HF disturbance
Bus probing	375	100.00 detection	Distinct spectral shift

Volume 1 • Issue 1 • Year 2026
ISSN: XXXX-XXXX | DOI:

5.3 Feature Importance and Engineering Interpretation

Random-forest feature-importance values show that high-frequency energy is the dominant descriptor, followed by spectral entropy, derivative variation, crest range, and lag-1 autocorrelation. This ordering is consistent with the threat model. Bus probing and clock faults introduce additional energy outside the narrow low-frequency burst pattern expected during normal operation, whereas transient faults alter slope and crest-to-crest characteristics. The importance profile also explains why a detector based only on mean power would fail in many cases: several modeled attacks preserve average power while changing spectral, temporal, or derivative properties.

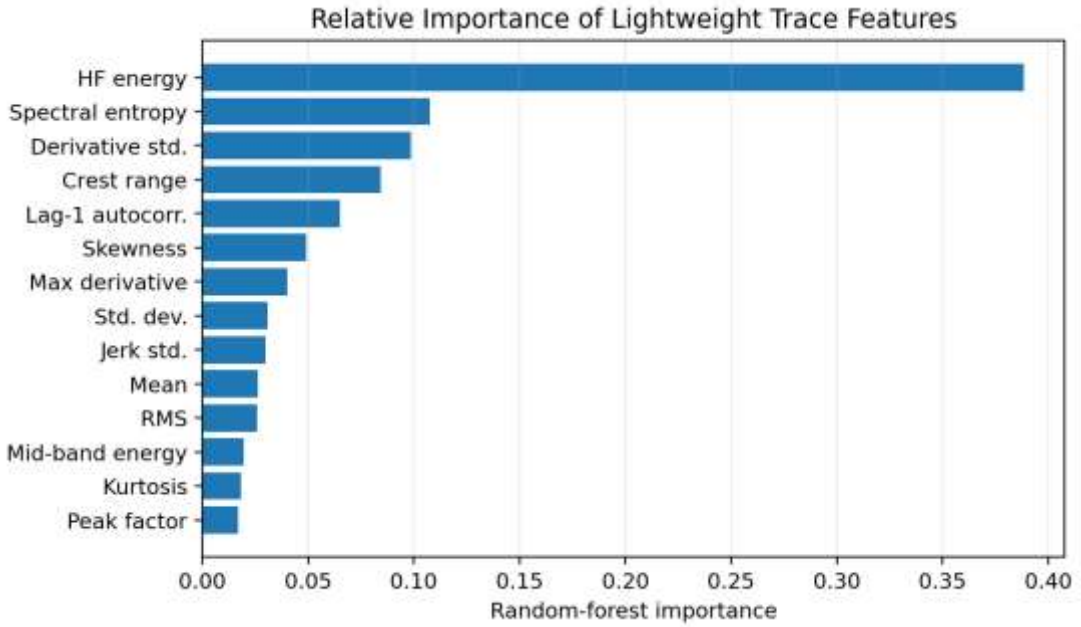

Fig. 3: Random-forest feature importance showing the dominant contribution of high-frequency energy and spectral-shape descriptors.

6. PRACTICAL DEPLOYMENT CONSIDERATIONS

6.1 Runtime Architecture

A practical implementation can be placed next to an accelerator power monitor, current sensor, voltage telemetry channel, or other side-channel proxy. Each inference window is buffered, normalized, transformed into the 14-feature vector, and evaluated by the classifier. One suspicious window is not sufficient to trigger a shutdown because benign transients are possible. Instead, the detector can require temporal persistence, such as two alerts within three consecutive inferences, before escalating to a secure response. Appropriate responses could include repeating the inference, transitioning into a trusted mode, verifying model integrity, reducing accelerator frequency, or logging the event for forensic analysis.

6.2 Computational Cost

The feature extractor is compact in terms of operations: most features require O(N) time, while the spectral descriptors require one 256-point real FFT, which is O(N log N) with a small constant factor. The machine-learning stage uses only 14 inputs. The Python evaluation times in Table 2 are intended for relative comparison of algorithmic complexity and should not be interpreted as embedded-hardware timing benchmarks, because the Python implementation is not optimized. Logistic regression is the simplest classifier, while RBF-SVM is more complex because it must evaluate support vectors. If an FPGA or microcontroller implementation requires lower latency, the weighted ensemble could be replaced with logistic regression combined with a linear or tree-based classifier, at the cost of some detection performance.

6.3 Limitations

The most significant limitation is that the traces are synthetic and therefore capture modeled rather than device-measured variability. They do not reflect board-level regulator dynamics, sensor bandwidth, electromagnetic coupling, quantization, temperature hysteresis, or device-to-device variation. Thus, while the results demonstrate algorithmic viability in a controlled setting, they do not establish protection performance for a production accelerator. Second, the explored attack parameters are illustrative rather than exhaustive, and an adaptive attacker could target the modeled feature distribution directly. Third, the probability weights were fixed for this dataset and would require retuning for a different processor or sensing modality. These limitations motivate the next steps: collecting labelled traces from FPGA, GPU, and microcontroller inference platforms and assessing cross-device generalizability.

7. CONCLUSION

This paper presents a lightweight hybrid framework for the detection of hardware-security anomalies in edge-AI accelerator traces. A reproducible simulation produced 12,000 short, power-like traces encompassing typical inferences, Trojan activations, voltage glitching, clock glitching, and bus-probing activities. Fourteen interpretable features in time-domain and spectral representations were used to train four traditional machine-learning models and a weighted probability ensemble. The proposed hybrid achieved 96.57% test accuracy, 98.01% precision, 95.07% recall, a 96.51% F1-score, 98.07% specificity, and a 0.9939 AUC. The results suggest that high-frequency energy, spectral entropy, derivative statistics, and autocorrelation yield discriminative information without the need for raw-trace deep learning.

From the perspective of this engineering study, the framework is intended as one layer in a broader security architecture, where compact side-channel monitoring complements cryptographic protection, trusted execution environments, and model-integrity checks. The approach is particularly relevant to edge computing because of its compact feature space and use of conventional machine-learning models. Further research should replace the synthetic trace generator with synchronized measurements from actual hardware, evaluate robustness against previously unseen attack vectors, study temperature and load drift, and investigate hardware-assisted feature extraction and adaptation. Until the results are reproduced under such conditions, they should be interpreted as a proof-of-concept demonstration rather than evidence of production-ready detection accuracy.

REFERENCES

1. L. Batina, S. Bhasin, D. Jap, and S. Picek, “CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel,” in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 515-532.

2. X. Hu et al., “DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints,” in Proc. 25th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2020, pp. 385-399. DOI: 10.1145/3373376.3378460.
3. M. Yan, C. W. Fletcher, and J. Torrellas, “Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures,” in 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 2003-2020.
4. S. Maji, U. Banerjee, and A. P. Chandrakasan, “Leaky Nets: Recovering Embedded Neural Network Models and Inputs Through Simple Power and Timing Side-Channels—Attacks and Defenses,” IEEE Internet of Things Journal, vol. 8, no. 15, pp. 12079-12092, 2021. DOI: 10.1109/JIOT.2021.3061314.
5. A. S. Rakin, Z. He, and D. Fan, “Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search,” in Proc. IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
6. F. Yao, A. S. Rakin, and D. Fan, “DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips,” in 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 1463-1480.
7. A. S. Rakin, M. H. I. Chowdhuryy, F. Yao, and D. Fan, “DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories,” in 2022 IEEE Symposium on Security and Privacy, 2022, pp. 1157-1174. DOI: 10.1109/SP46214.2022.9833743.
8. Z. Zhan, Z. Zhang, S. Liang, F. Yao, and X. Koutsoukos, “Graphics Peeping Unit: Exploiting EM Side-Channel Information of GPUs to Eavesdrop on Your Neighbors,” in 2022 IEEE Symposium on Security and Privacy, 2022, pp. 1440-1457. DOI: 10.1109/SP46214.2022.9833773.
9. L. Breiman, “Random Forests,” Machine Learning, vol. 45, pp. 5-32, 2001.
10. C. Cortes and V. Vapnik, “Support-Vector Networks,” Machine Learning, vol. 20, pp. 273-297, 1995.
11. J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” The Annals of Statistics, vol. 29, no. 5, pp. 1189-1232, 2001.
12. F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 413-422.
13. P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” in Advances in Cryptology—CRYPTO ’99, Lecture Notes in Computer Science, vol. 1666, 1999, pp. 388-397.
14. R. Méndez Real and R. Salvador, “Physical Side-Channel Attacks on Embedded Neural Networks: A Survey,” Applied Sciences, vol. 11, no. 15, Art. 6790, 2021. DOI: 10.3390/app11156790.