

LEAF-NIDS: A Lightweight Ensemble Architecture for Feature-Efficient Network Intrusion Detection at the Edge

Jai Mungi, Harsh Mathur
Dept. of Computer Science & Engineering
Rabindranath Tagore University (RNTU)
Bhopal, India

ABSTRACT

Stacking-based ensemble models with wrapper-based feature selection have proven to provide high accuracy in intrusion detection, but the models can be very computationally intensive, restricting their use in resource-constrained edge and IoT environments. This paper introduces a lightweight network intrusion detection system framework that uses adaptive feature selection (AEFSS) and has resource efficiency for use in gateway-class edge devices, called LEAF-NIDS. LEAF-NIDS takes away the need for iterative wrapper search and applies a single-pass Symmetrical Uncertainty-based redundancy-aware (SU-R) feature selection approach, using a cascaded weighted voting ensemble of a depth-bounded decision tree, Naïve Bayes, and a linear classifier. To enhance deployment efficiency, the framework also supports 8-bit quantization and branch pruning to compress the model after training, which further minimizes the model size and inference time. The suggested architecture is designed to be deployed on PoC lightweight edge devices such as NVIDIA Jetson Nano and Raspberry Pi 4. A theoretical analysis of computational complexity and resource footprints shows the efficiency of LEAF-NIDS over representative wrapper-based and stacking-based approaches. Furthermore, a comprehensive evaluation methodology with benchmark datasets, deployment platforms, performance metrics, and comparative baselines is presented, allowing for systematic evaluation. The proposed framework offers a practical compromise between detection performance, computational efficiency, and deployment feasibility and can be well adapted for real-time edge-based network intrusion detection.

General Terms

Security, Algorithms, Performance

Keywords

Intrusion Detection System, Lightweight Machine Learning, Ensemble Learning, Edge Computing, Knowledge Distillation, IoT Security.

1. INTRODUCTION

Network Intrusion Detection Systems (IDSs) are becoming even more integral to a well-provisioned datacenter back-end, and also to the network edge — the gateway devices, industrial controllers and IoT hubs between untrusted traffic and the assets being protected. IDS: A security system, implemented in software, hardware or both, that actively scans network traffic or host activity to detect unauthorized access or misuse or malicious activity, and notifies the administrator or initiates a response when an issue is detected. IDSs can be broadly categorized according to three criteria: detection scope (monitoring at the host level, or network level; network level refers to monitoring traffic that passes through a network

segment); architectural design (centralized, decentralized or distributed); and detection technique (signature-based, which compares the observed traffic with a database of known attacks; anomaly-based, which identifies deviations from a learned model of acceptable traffic; or hybrid, which combines both techniques to achieve a balance between detection coverage and false-alarm rate) [1, 2]. The overall structure or architecture of most IDS deployments, regardless of the actual detection algorithm, is shown in Fig. 1: The sensors gather host or network traffic, perform a preprocessing operation to reduce the data to a feature representation, feed it into the detection engine, and finally into the decision logic, which may result in alerts being generated to be consumed by the security administrator or sent to an SIEM console for processing, and into a signature or rule that can be returned to the detection engine for further processing. The framework suggested in this chapter is most easily viewed as a resource-efficient implementation of the feature-extraction and detection-engine phases of this general framework, designed to run on a constrained gateway hardware platform.

The hybrid intrusion detection systems based on correlation-based filtering and wrapper-based search and stacking ensembles have achieved good detection accuracy on benchmark data sets, such as NSL-KDD and UNSW-NB15. The costs of the wrapper phase of pipelines like this mean that many small subsets of features are tested repeatedly during cross-validation, and the stacking phase involves training and storing multiple base learners and a meta-classifier something that costs little in a data-center training pipeline, but is hard to justify on a device with a few hundred megabytes of RAM, and none attached to a GPU. Several recent efforts have directly tackled this problem by introducing stacking ensembles with feature-importance pruning, feature-selection pipelines that are quantized and pruned for deployment at the edge, and knowledge-distillation techniques that convert deep IDS models into small student networks that operate on microcontroller- and single-board-computer-class devices. All of these efforts are united by a single theme: evaluating detection accuracy for memory footprint, inference latency, and energy consumption, especially as it relates to gateway devices that have limited power and compute resources and must classify traffic in real-time within a fixed budget.

In this work, a LEAF-NIDS (Lightweight Ensemble Architecture for Feature-efficient Network Intrusion Detection System), which is built with the joint goal in mind has been proposed. LEAF-NIDS does not require that the feature-selection stage be adapted after the fact for a heavier hybrid pipeline, nor does it require that the classification stage be adapted after the fact to be run in a single pass on a cascade the way that traditional single-pass classifiers are configured. Instead, LEAF-NIDS restructures both of these stages, based

on single-pass, cascade-friendly operations, and adds an explicit post-training compression stage, where the resource

cost is a first-class design constraint, not an afterthought.

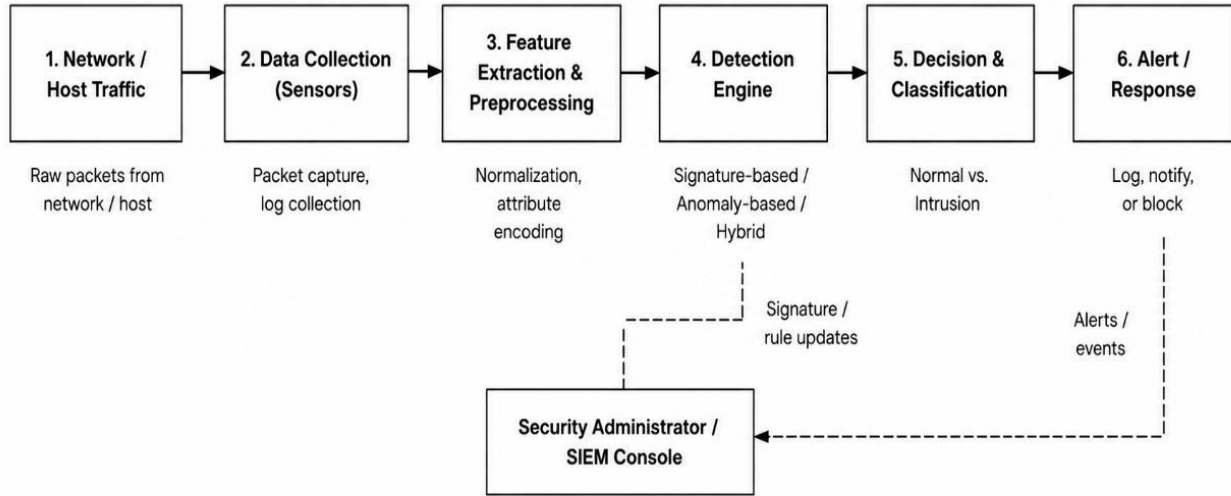


Fig 1: General architecture of a Network Intrusion Detection System.

The contributions of this paper are:

- A single-pass, filter-only feature selection procedure (SUR) which eliminates the iterative wrapper search in heavier hybrid pipelines while maintaining a built-in redundancy-control mechanism that reduces the cost of feature-selection due to repeated-fold wrapper evaluation to a single pass of ranking and pruning.
- The cascaded ensemble model is a lightweight ensemble model which is equivalent to a stacking model, but instead of a stacking model, a cheap Tier-0 gate is used to dispose of most of the benign traffic flows, and only the suspicious-anomalous traffic flows are passed on to the heavier category-, name-level classifiers.
- An integrated post-training compression stage (quantization and branch pruning) as an embedded processing stage rather than a deployment step.
- A theoretical complexity and memory-footprint analysis of LEAF-NIDS with respect to the wrapper-based and stacking-based baselines, along with a complete experimental protocol, datasets, and target hardware and metrics for an empirical validation to follow in a later paper.

2. RELATED WORK

2.1 Lightweight and Edge-Oriented Intrusion Detection

There are a number of recent studies that focus on the deployment of IDS directly on constrained hardware. In this work, the authors proposed an optimized IDS system for edge-based IIoT security by utilizing a hybrid technique of Particle Swarm Optimization feature selection along with dimensionality reduction (PCA, LDA, t-SNE). The authors reported that dimensionality reduction and quantization of the model into 8-bit after training reduced the inference time by about 25% in a simulated edge environment. Abdulkareem et al. proposed a lightweight stacking ensemble learning (SEL) based feature importance dimensionality reduction for the attack detection of IIoT, achieving a 86.9% reduction in the storage requirements with high multiclass accuracy [4]. Hasan et al. suggested an autoencoder-based feature-learning pipeline

for IIoT intrusion detection specifically designed for portability while keeping the deployment budget light [5]. To address the energy and latency challenges of edge computing, Umar et al. proposed an energy-efficient deep-learning IDS (DNN-KDQ) specifically tailored for edge computing [6]. Gao et al. suggested a low computation-per-inference multiclassification IDS for edge IoT networks, which is able to maintain multiclass discrimination with low computation per inference [7]; Lundqvist et al. have compared a number of lightweight machine-learning models for on-device IoT IDS with realistic resource limits [8]. Misrak and Melaku implemented the advanced feature engineering and dynamic quantization to develop a compact IDS for IoT gateways [9]. Additional ensembles approaches for edge deployment have been studied, such as an ensemble technique for IDS at the edge of IoT systems that aims at a tradeoff between detection and computational complexity [10] and a hybrid machine-learning based IDS for edge deployment of the IIoT systems, which has a focus on deployability rather than marginal accuracy improvement [21]. The accuracy of transformer models (e.g., transfer-learning models proposed by Ullah et al. on imbalanced network traffic [11]) is good, but there is a high computation cost that the authors themselves have stated is prohibitive for limited deployment.

2.2 Model Compression and Knowledge Distillation for IDS

A parallel line of work shortens bigger IDS models, as opposed to building lighter IDS models from scratch. Benaddi et al. used SHAP-guided feature pruning together with knowledge-distilled Kronecker-structured student networks, achieving a student model under three orders of magnitude smaller than its teacher and its macro-F1 score over 0.986 on the TON_IoT dataset [12]. In [13] Wisanwanichthan and Thammawichai showed that they can compress deep IDS models to use with a UAV and IoT system, retaining almost as much detection capability as the teacher while using a very small amount of resources. The self-knowledge-distillation approach for network intrusion detection proposed by Yang et al. eliminates the need for a pre-trained teacher network entirely, saving the network from being compressed to less than 1 MB without compromising the testing accuracy of BERT-like models on

NSL-KDD and IoT-23 [14]. They are based on the basic knowledge-distillation formulation proposed by Hinton et al. [15] and the more general insight gained in this original paper that an ensemble’s knowledge can typically be distilled into a single, significantly smaller model without a commensurate loss of accuracy.

2.3 Ensemble Learning Trade-offs at the Edge

Ensemble learning is still appealing for IDS as it decreases the variance of each class which is especially useful for those classes that are under-represented, but classical ensemble and stacking constructions, such as Random Forest [16] as a variance-reduction mechanism or stacked generalization for heterogeneous base learners, were not developed with a resource budget in mind. Recent light-weight ensemble studies indicate that this trade-off can be actively managed, for instance, by shrinking the ensemble, limiting the depth of the trees or by switching the learned meta-classifier with a less expensive fixed or lightly adaptive combination rule; a systematic design of this type that integrates feature selection, cascade style inference gating and post-training compression into a single coherent pipeline, however, is comparatively rare and should be evaluated not only against the raw deep-learning

baselines but against the wrapper-and-stacking baselines as well. The gap encourages the design of LEAF-NIDS.

3. PROPOSED LEAF-NIDS FRAMEWORK

3.1 Design Rationale

The main inspiration behind LEAF-NIDS comes from the contrast with heavier hybrid pipelines which feature correlation-based filtering, iterative wrapper search (evaluating multiple appraiser classifiers over repeated iterations on each cross-validation fold) and use a trained meta-classifier to combine multiple base classifiers. The first one works well for maximizing accuracy, but is highly compute and memory intensive in the presence of many candidate subsets to search, and the second one is compute and memory intensive because each base learner’s output must be computed and stored for each instance, even for trivially benign ones, for use in the stacking meta-classifier. LEAF-NIDS is designed in such a way that (i) feature selection runs in a single pass over the training set of features, (ii) most of the traffic (normal, high-confidence traffic) is sent to a cheapest possible stage, and (iii) the deployed artifact is explicitly compressed instead of deployed at full training-time precision. Overall architecture is shown in Fig. 2.

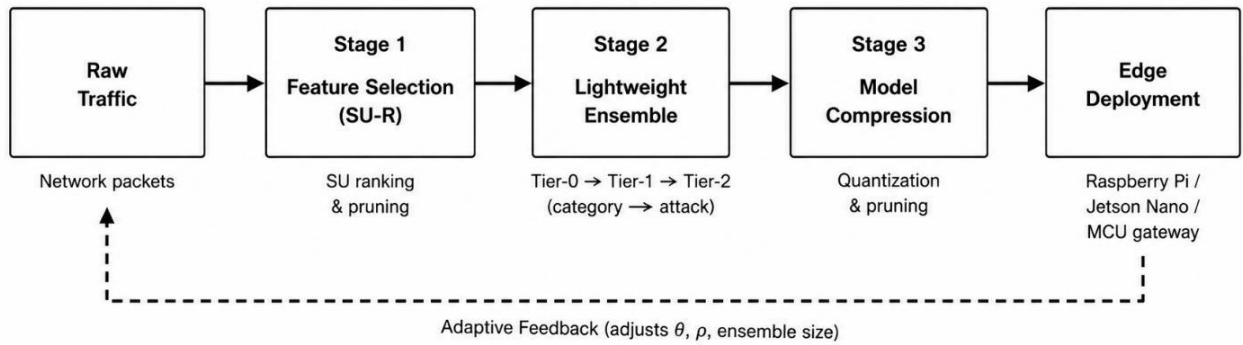


Fig 2: Overview of the proposed LEAF-NIDS architecture.

3.2 Stage 1: Single-Pass Lightweight Feature Filtering (SU-R)

Like correlation-based filtering in general, Symmetrical Uncertainty (SU) is employed to measure the statistical dependency between each feature f_i and the class label C , and normalize MIF to prevent it from being biased toward high-cardinality features:

$$SU(f_i, C) = 2 \times [IG(f_i | C) / (H(f_i) + H(C))] \quad (1)$$

where $IG(f_i | C)$ is the information gain of f_i with respect to C and $H(\cdot)$ denotes Shannon entropy. Unlike hybrid pipelines that treat SU-filtered candidates as input to a subsequent wrapper search, SU-R (Symmetrical-Uncertainty with Redundancy control) performs feature selection in a single pass:

- Rank all features by $SU(f_i, C)$ in descending order.
- Initialize the selected set $S = \{\}$.
- For each feature f_i in ranked order: if $SU(f_i, C) < \theta$ (relevance threshold), stop — all remaining features are less relevant and are discarded. Otherwise, compute the maximum pairwise SU between f_i and any feature already in S ; if this redundancy score exceeds a redundancy threshold ρ , discard f_i as redundant. Otherwise, add f_i to S .

- Return S .

This is a single greedy pass at cost $O(n \log n)$ for the initial ranking, plus a cost of $O(|S| \cdot n)$ to check for redundancy of selected features with the classifiers in the appraiser, against multiple k-fold cross-validation repeated runs for both forward and backward wrapper search and the best-first. The price of this is clear: SU-R does not improve a locally suboptimal ranking-based decision as a wrapper search can, and so it is expected that it will be more likely to select a slightly suboptimal subset, but with a huge constant factor reduction in the selection cost (which is consistent with the observations of previous filter-vs-wrapper comparisons).

3.3 Stage 2: Cascaded Lightweight Ensemble Classification

LEAF-NIDS is designed to structure classification as a confidence-gated cascade to train a stacking meta-classifier over the output of each base learner for each instance instead of training a single stacking meta-classifier on the output of multiple base learners for each instance (Fig. 3):

- **Tier-0 gate.** Stump-based decision trees with only a single depth ($d = 2$) trained to classify normal traffic from suspected anomalies at a very low computational cost.

Traffic that is given a high-confidence normal classification exits the pipeline immediately, and traffic that is classified as low-confidence anomalous goes to Tier-1.

- **Tier-1 weighted-vote ensemble.** Each voted on the attack category (DoS, Probe, R2L, U2R) based on three cheap, heterogeneous base learners: shallow decision tree (J48, depth ≤ 3), Naive Bayes, and a linear classifier. Weighted majority voting is used to combine votes by assigning weights to the votes of each learner according to its held-out validation accuracy and setting the weights periodically instead of learning them through a trained meta-classifier. This avoids the need to maintain and evaluate a fourth (meta-)model for each instance.

- **Tier-2 category-specific mini-ensemble.** After a coarse attack category is assigned, a small category-specific ensemble (which consists of the same three types of base-learners, but is retrained only on instances of the one category) resolves the fine-grained attack name, and achieves the same level of granularity in the forensic analysis as heavier two-tier frameworks, and without having to keep a single large multi-class ensemble in memory simultaneously. The expected amortized inference cost per instance is dominated by the cost of the cheap gate and not the full ensemble, as the full ensemble is only called on the (usually minority) set of traffic that is anomalous or ambiguous.

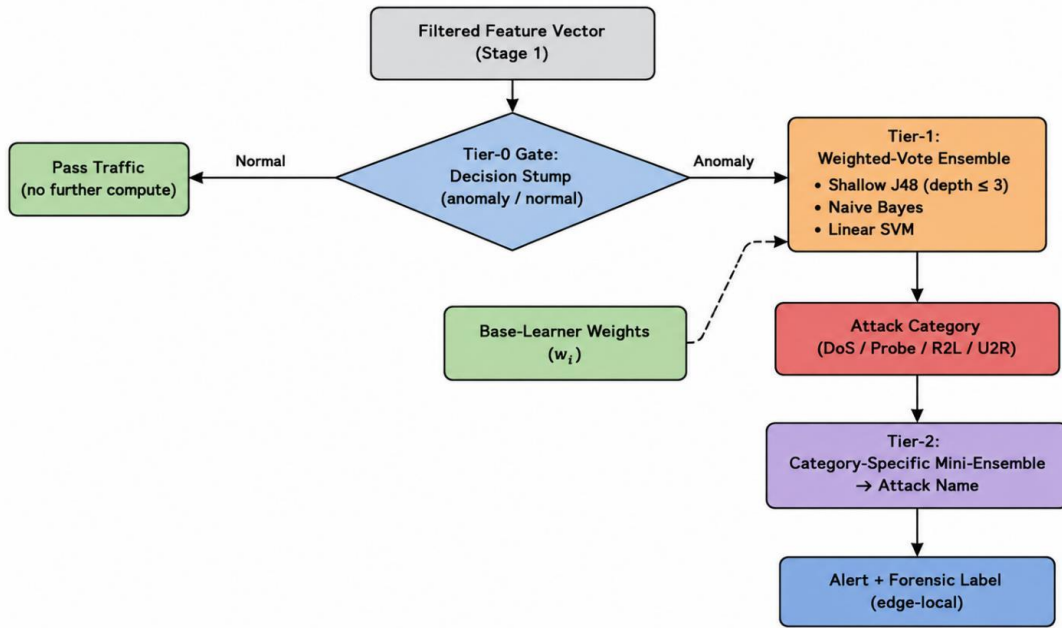


Fig 3: Cascaded lightweight ensemble decision flow (LEAF-NIDS Stage 2).

3.4 Stage 3: Post-Training Compression comes after the training phase.

Following the training of the cascade, LEAF-NIDS performs two compression steps before deployment, as described in the quantization and pruning processes described in the lightweight IDS literature [3, 9]:

- **Weight/threshold quantization.** Inferring with quantized models for integer arithmetic has been the standard way of quantizing trained models, and the same approach is employed to quantize floating-point split thresholds and Naive Bayes probability tables.
- **Low-importance branch pruning.** Branches of the decision tree that do not contribute significantly to the validation accuracy are pruned, reducing the size of the stored tree without affecting its accuracy significantly (similarly to what knowledge-distillation-based compression does, except that it is applied directly to the decision tree ensemble, and does not require a separate teacher–student training procedure).

The artifact deployed to the edge device is the compressed model, not the full-precision training-time model, and all complexity and footprint figures reported in Section 4 are for this compressed artifact.

3.5 LEAF-NIDS Algorithm

Algorithm 1 outlines the entire process.

Algorithm 1: LEAF-NIDS Training and Compression

Input: Dataset D , features F , class C , relevance threshold θ , redundancy threshold ρ

Output: Deployable compressed cascade model M

- 1: Preprocess D : min-max normalization, integer encoding of nominal attributes
 - 2: $S \leftarrow \emptyset$
 - 3: Rank F by $SU(f_i, C)$ descending
 - 4: **for each** f_i in ranked F **do**
 - 5: **if** $SU(f_i, C) < \theta$ **then break**
 - 6: **if** $\max_{f_j \in S} SU(f_i, f_j) > \rho$ **then continue** // redundant, skip
 - 7: $S \leftarrow S \cup \{f_i\}$
 - 8: **end for**
 - 9: Train Tier-0 decision stump on S (binary: normal vs. anomaly)
 - 10: Train Tier-1 base learners {J48 (depth ≤ 3), NaiveBayes, LinearClf} on S
 - 11: Compute Tier-1 weights w_i proportional to validation accuracy of learner i
 - 12: **for each** attack category k **do**
 - 13: Train Tier-2 mini-ensemble on category- k instances (same learner types)
 - 14: **end for**
 - 15: Quantize all thresholds/probability tables to 8-bit fixed-point
 - 16: Prune tree branches with importance below pruning threshold
 - 17: $M \leftarrow \{\text{Tier-0, Tier-1 (with weights), Tier-2 ensembles}\}$
 - 18: **return** M
-

4. THE DEPLOYMENT ARCHITECTURE AND COMPLEXITY ANALYSIS OF EDGES

The proposed LEAF-NIDS framework will be implemented on the gateway-class edge devices that sit at the edge of an untrusted network segment and the assets it serves. Representative deployment platforms are the Raspberry Pi 4, or NVIDIA's Jetson Nano, or other ARM Cortex-A/M-class embedded systems. The SU-R feature selection, cascaded lightweight ensemble (Tier-0 to Tier-2), and the compressed inference model are meant to be placed in on-device memory and thus do not require cloud-assisted inference for normal packet classification. Communication overhead is reduced since only summary statistics, like periodic updates of Tier-1 weights and aggregate detection counts, are communicated to a central monitoring server; other than this, low-latency intrusion detection is maintained. This deployment approach is in line with previous edge-based IDS research that successfully quantizes models for deployment on Jetson Nano-class devices and achieves sub-millisecond inference latency after quantization and on-device deployment [3, 4, 9].

To evaluate the feasibility of deployment, LEAF-NIDS is theoretically evaluated against a representative baseline that consists of a wrapper and stacking ensembles with a general

architecture similar to CFS-based feature selection, wrapper optimization, and stacking ensembles. The comparison is based on computational complexity, model storage, and deployment features, not experimental performance. The LEAF-NIDS significantly cuts down on the computational cost by using a lightweight ensemble processing suspicious traffic (a cascaded approach), a single-pass SU-R feature selection algorithm replacing iterative wrapper search, and post-training quantization and pruning to minimize the model size, as demonstrated in Table 1. Such design decisions enhance computational efficiency and memory usage, rendering such a system readily applicable to resource-constrained environments on the edge as compared to traditional wrapper-based systems.

One of LEAF-NIDS' central design goals is to achieve a trade-off between intrusion detection and computational efficiency. Feature selection and stacking ensembles can provide small gains in prediction accuracy, but at significant additional training, inference, and storage costs. On the other hand, the proposed framework focuses on lightweight computation and minimal deployment footprint and has comparable detection performance with much lower computational costs, as seen in the literature of lightweight IDSs recently, which shows that the compressed and cascade-based models have similar detection capabilities with significantly lower computational costs [3, 4, 9].

Table 1. Theoretical Complexity Comparison

Stage	Wrapper + Stacking Baseline	LEAF-NIDS
Feature selection	Iterative forward/backward/best-first search; each candidate subset evaluated under k-fold CV with 2 appraiser classifiers → cost scales with number of candidate subsets explored × k	Single ranking pass $O(n \log n) + O(S \cdot n)$ redundancy check; no repeated CV evaluation
Base learners resident at inference	All base learners evaluated for every instance, plus a trained meta-classifier	Tier-0 gate evaluated for every instance; Tier-1/Tier-2 only evaluated for the minority of instances flagged anomalous
Model storage	Full-precision trees, NB tables, meta-classifier weights	8-bit quantized thresholds/tables; pruned tree branches
Expected deployment footprint	Multiple MB (typical for multi-learner stacking ensembles at full precision)	Sub-MB to low single-digit MB range, consistent with reported quantized/pruned lightweight IDS artifacts [3], [4], [9]

5. EXPERIMENTAL EVALUATION METHODOLOGY & DISCUSSION

The proposed LEAF-NIDS framework is tested against the benchmark intrusion detection datasets to demonstrate the effectiveness of detecting intrusions and the efficiency of deployment. NSL-KDD (20% stratified subset with 41 features) and UNSW-NB15 are chosen as the major benchmark dataset for continuity with the previous IDS research, and ToN_IoT and Edge-IIoTset are used for further evaluation in the IoT and edge-computing conditions, following recent lightweight IDS research [4], [5], and [12] respectively.

The deployment targets are representative edge computing platforms, including the Raspberry Pi 4 (ARM Cortex-A72, CPU-only) and the NVIDIA Jetson Nano, enabling evaluation of the platform in both CPU-based and lightweight GPU-assisted environments. These platforms are popular for lightweight IDS research and are useful for simulating deployment on an edge resource-limited device.

Performance assessment takes into account detection capability

and computational efficiency. Common classification measures are accuracy, True Positive Rate (TPR), False Positive Rate (FPR), F1-Score, Matthews Correlation Coefficient (MCC), and Area Under the ROC Curve (AUC-ROC); the latter measure gives extra robustness with class-imbalanced intrusion datasets [20]. Resource-oriented metrics are introduced, such as feature-selection time, per-instance inference latency, deployed model size, memory footprint and energy consumption per inference to evaluate the framework comprehensively for its use on the edge.

As done in recent lightweight edge-IDS literature, reporting the utilization of computational resources in addition to detection accuracy is standard practice. The 3 types of baseline approaches are:

- representative wrapper-plus-stacking hybrid frameworks,
- standalone filter-based feature selection methods, such as CFS, and
- recently published lightweight or compressed intrusion detection models discussed in Section 2.

This comparison gives a fair evaluation of the detection

performance/computational efficiency trade-off. In order to further explore the contribution of individual components, the framework is tested under a number of configurations by means of an ablation analysis. These include removing the Tier-0 filtering stage to measure its contribution to inference efficiency, adjusting the redundancy threshold (ρ) of the SU-R feature selection algorithm to understand the trade-off between the number of features and detection accuracy, and testing the model before and after post-training quantization and pruning to highlight the effect of model compression on the model's predictive power and deployment footprint.

While LEAF-NIDS is built to be lightweight and deployed at the edge, there are several design considerations. The single-pass SU-R feature selection algorithm uses a greedy algorithm based on removing redundant features and does not necessarily find complex relationships between features as well as those found by more complex algorithms such as wrapper-based. Also, the performance of the cascaded architecture is tied to the selection of a correct Tier-0 confidence threshold; too "conservative" could lead to more samples being passed to the following tiers, which would result in lower computational savings. Moreover, as quantization and pruning are applied after training, it is important to carry out a class-wise evaluation for the detection of minority attack categories (e.g., U2R and R2L) apart from the overall performance. Notwithstanding these factors, the proposed framework successfully achieves an optimum balance between detection capability, computational complexity, and deployment efficiency, which is suitable for resource-constrained edge-based intrusion detection systems.

6. CONCLUSION AND FUTURE WORK

The proposed LEAF-NIDS is a lightweight ensemble architecture for network intrusion detection that redefines feature selection as a one-pass filter-only procedure, redefines classification as a cascade of cheap base learners, and explicitly represents post-training quantization and pruning as a deployment stage. The design explicitly targets the edge area of gateway-class hardware and is thought of as a conscious accuracy-for-resource-efficiency compromise when compared to more weighty pipelines wrapped with a hybrid stacking pipeline. In this paper, no experimental results are claimed, but the next step in this thesis, which is the empirical validation of LEAF-NIDS, is provided with a complete evaluation protocol, including datasets, target hardware, metrics, baselines, and ablations.

Experimental validation is just the first step in future work that would go beyond the immediate proposed application; further research would involve expanding the cascade to a federated scenario in which the weights of Tier-1/Tier-2 are updated collectively across gateway devices without centralized storage of raw traffic data and examining whether the SU-R filter can be adapted online when traffic distributions change at the edge.

7. REFERENCES

- [1] R. Bace and P. Mell, "NIST Special Publication on Intrusion Detection Systems," NIST, Gaithersburg, MD, USA, Tech. Rep. 800-31, 2001.
- [2] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Chalmers Univ. of Technology, Goteborg, Sweden, Tech. Rep. 99-15, 2000.
- [3] R. S. Tiwari, D. Lakshmi, T. K. Das, A. K. Tripathy, and K.-C. Li, "A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security," *Telecommun. Syst.*, vol. 87, pp. 605–624, 2024, doi: 10.1007/s11235-024-01200-y.
- [4] S. A. Abdulkareem, C. H. Foh, F. Carrez, and K. Moessner, "A lightweight SEL for attack detection in IIoT networks," *J. Netw. Comput. Appl.*, vol. 230, Art. no. 103980, 2024, doi: 10.1016/j.jnca.2024.103980.
- [5] T. Hasan, A. Hossain, M. Q. Ansari, and T. H. Syed, "Enhanced intrusion detection in IIoT networks: A lightweight approach with autoencoder-based feature learning," in *Proc. Int. Conf. Internet of Things, Big Data and Security*, 2025, arXiv:2501.15266, doi: 10.48550/arXiv.2501.15266.
- [6] H. G. A. Umar et al., "Energy-efficient deep learning-based intrusion detection system for edge computing: a novel DNN-KDQ model," *J. Cloud Comput.*, vol. 14, Art. no. 32, 2025, doi: 10.1186/s13677-025-00762-9.
- [7] W. Gao, M. Wang, Y. Pei, F. Li, and C. Wang, "A lightweight multiclassification intrusion detection model for edge IoT networks," *Electronics*, vol. 15, no. 5, Art. no. 938, 2026, doi: 10.3390/electronics15050938.
- [8] J. Lundqvist, A. Hadzic, T. M. Kirkeluten, and M. P. N. Halkjelsvik, "Lightweight machine learning models for intrusion detection on IoT devices," *Norsk IKT-konferanse for forskning og utdanning (NIKT)*, vol. 37, no. 3, 2025, doi: 10.5324/jrxdbj92.
- [9] S. F. Misrak and H. M. Melaku, "Lightweight intrusion detection system for IoT with improved feature engineering and advanced dynamic quantization," *Discover Internet of Things*, vol. 5, Art. no. 97, 2025, doi: 10.1007/s43926-025-00203-8.
- [10] A. Aldaej, I. Ullah, and M. Atiquzzaman, "Ensemble technique of intrusion detection for IoT-edge platform," *Sci. Rep.*, vol. 14, Art. no. 11703, 2024, doi: 10.1038/s41598-024-62435-y.
- [11] F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, "IDSInt: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic," *Digit. Commun. Netw.*, vol. 10, no. 1, pp. 190–204, 2024, doi: 10.1016/j.dcan.2023.03.008.
- [12] H. Benaddi, M. Jouhari, N. Laamech, A. Motii, and K. Ibrahim, "Lightweight intrusion detection in IoT via SHAP-guided feature pruning and knowledge-distilled Kronecker networks," arXiv:2512.19488, 2025.
- [13] T. Wisanwanichthan and M. Thammawichai, "A lightweight intrusion detection system for IoT and UAV using deep neural networks with knowledge distillation," *Computers*, vol. 14, no. 7, Art. no. 291, 2025, doi: 10.3390/computers14070291.
- [14] S. Yang, X. Zheng, Z. Xu, and X. Wang, "A lightweight approach for network intrusion detection based on self-knowledge distillation," arXiv:2307.10191, 2023.
- [15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.
- [16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [17] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.
- [18] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Communications and*

Information Systems Conf. (MilCIS), Canberra, Australia, 2015, pp. 1–6.

- [19] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA), Ottawa, Canada, 2009, pp. 1–6.

- [20] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score

and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, Art. no. 6, 2020.

- [21] H. Yao, P. Gao, P. Zhang, J. Wang, C. Jiang, and L. Lu, “Hybrid intrusion detection system for edge-based IIoT relying on machine-learning-aided detection,” *IEEE Netw.*, vol. 33, no. 5, pp. 75–81, 2019, doi: 10.1109/MNET.001.1800479.