IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151111

Lightweight Machine Learning for DoS Attack Detection in Wireless Sensor Networks: An Implemented Decision Tree System

Prof. Shahid Raza[1], Lalit Patil[2], Sachin Chavan[3], Tejaswini Shirsath[4], Shreyas Ranshoor[5]

Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India[1-5]

Abstract: Wireless sensor networks require intrusion detection methods that respect limited memory, processing capacity, bandwidth, and battery supply. This paper presents an implemented multiclass Denial-of-Service (DoS) attack detection system for Wireless Sensor Networks using the WSN-DS dataset and lightweight machine-learning models. The public prototype exposes manual prediction and bulk CSV analysis through a React front end and a Flask API. The application reports 374,661 dataset records, 17 input features, five attack classes, and a headline accuracy of 99.47%. Its model-comparison dashboard reports the highest displayed accuracy for a 17-feature Random Forest at 99.72%; however, the Random Forest is not the most time- or memory-efficient option. The study evaluates the full 17-feature representation and also selects a reduced Decision Tree using the five highest ranked features: *Is CH, ADV S, SCH S, Data Sent To BS, and Expanded Energy*. The five classes are Normal, Flooding, Blackhole, Grayhole, and TDMA. The selected five-feature Decision Tree provides the intended balance of accuracy, training efficiency, model size, interpretability, and reduced feature processing overhead for a lightweight WSN detector. Training-time and memory values are reported as displayed by the dashboard and are not interpreted as packet level inference latency or direct energy measurements. The study links a reproducible web deployment to a resource-aware WSN DoS detection design and identifies the measurements required for hardware-level validation.

Index Terms: Wireless sensor networks, denial-of-service detection, intrusion detection, lightweight machine learning, WSN-DS, decision tree, random forest.

I. INTRODUCTION

Wireless sensor networks (WSNs) operate under a different engineering constraint set from conventional servers and data-center networks. Sensor nodes collect measurements, exchange short wireless frames, and forward information toward a sink or base station while working with restricted memory, processor speed, radio bandwidth, and stored energy. These constraints make security design a systems problem: a detector that improves classification but consumes excessive node resources can shorten network lifetime or disrupt the sensing task it is intended to protect [4], [17], [19].

Denial-of-Service (DoS) traffic is particularly damaging because it targets availability. Repeated or malformed requests can occupy the channel, increase forwarding work, cause retransmissions, and reduce the opportunity for legitimate reports to reach the base station. Prior WSN security research has examined rule-based protection, cluster-level monitoring, statistical methods, classical classifiers, ensemble methods, and deep models [6], [9], [18]. The principal design question is not simply whether an algorithm detects an attack. It is whether the algorithm gives sufficient separation between benign and malicious traffic while remaining small enough for a constrained deployment.

This paper documents an implemented WSN intrusion-detection application and its public prototype, and places the resulting observations within the lightweight machine-learning literature. The system uses the WSN-DS dataset, evaluates several classical and ensemble model variants, and provides manual and CSV-based prediction interfaces. Its current output is multiclass: Normal, Flooding, Blackhole, Grayhole, and TDMA. Flooding is the class most directly associated with DoS behavior, while the remaining classes represent additional WSN attack categories. The central project decision is to select the Decision Tree as the best efficient model for deployment: the Random Forest achieves slightly higher displayed accuracy, but its training-time and memory requirements are substantially larger.

The contributions are fourfold:

- The paper records the architecture and observable behavior of a deployed WSN-DS intrusion-detection system.

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



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151111

- It compares the dashboard-reported accuracy, training time, and memory values for the visible model variants.
- It explains why a five-feature Decision Tree is a reasonable deployment choice even though the full-feature Random Forest has the highest displayed accuracy.
- It specifies a reproducible lightweight detection workflow that separates the completed implementation from measurements reserved for later hardware validation.

II. REVIEW METHODOLOGY

The review was organized around three linked questions: which methods have been used for WSN DoS detection; how lightweight classifiers compare with deep or ensemble alternatives; and which design choices support deployment on constrained nodes. Searches covered WSN intrusion detection, DoS and DDoS detection, WSN-DS, feature selection, lightweight machine learning, and ensemble learning. The collected literature included surveys, dataset papers, classical ML studies, deep-learning studies, and recent lightweight or selective-mitigation approaches [1], [2], [16], [21].

The review retained studies that addressed at least one of the following: WSN or closely related IoT intrusion detection; DoS, DDoS, or resource-exhaustion traffic; a classifier suitable for constrained devices; feature selection or dimensionality reduction; or an evaluation on a named dataset. Studies were compared by attack scope, learning method, feature handling, reported metrics, and deployment assumptions. Accuracy was not treated as a sufficient measure on its own because class imbalance, data leakage, training cost, memory footprint, and the difference between offline training and online inference can alter deployment suitability.

The implementation analysis used the public prototype interface and its model-comparison dashboard. The study first evaluates models using all 17 WSN-DS features and then evaluates reduced feature configurations, including a five-feature Decision Tree for lightweight deployment. The prototype reports the WSN-DS dataset, 374,661 records, 17 features, five attack classes, and a headline accuracy of 99.47%. Its dashboard reports eight tested models or feature-model variants, a best displayed accuracy of 99.72%, and a manual-deployment choice labelled “DT Top-5.” The manua l-prediction page requests the five selected features *Is CH, ADV S, SCH S, Data Sent To BS, and Expanded Energy*, while the CSV page provides bulk upload and attack-pattern analysis. Only values visible in the prototype were transcribed. The dashboard’s time column is labelled “Training Time”; it is therefore not interpreted as packet-level inference latency. Likewise, the memory column is reported as displayed and is not converted into a claim about battery energy.

III. THEMATIC LITERATURE REVIEW

A. Datasets and WSN attack representation

The WSN-DS dataset provides a common basis for comparing WSN intrusion-detection models and includes traffic associated with multiple attack categories [1]. Its reuse supports reproducibility, but a single benchmark cannot represent every radio protocol, topology, node density, mobility pattern, or attacker capability. Dataset construction and feature semantics therefore remain central to any claim about generalization.

The implemented system uses the complete set of 17 numerical WSN-DS features for the full-feature experiments and maps each record to one of five classes: Normal, Flooding, Blackhole, Grayhole, and TDMA. For the lightweight deployment configuration, the study reduces the input to five selected features and trains a Decision Tree. This multiclass framing is more informative than a single binary label because an operational response can differ by attack family. It also requires care in interpreting the term DoS. Flooding is the direct availability-oriented class, whereas Blackhole,Grayhole, and TDMA describe other forms of malicious or abnormal network behavior.

B. Classical lightweight classifiers

Decision Trees remain attractive for constrained environments because their inference path consists of a small sequence of feature comparisons. Quinlan’s C4.5 framework established a durable foundation for tree induction, and subsequent WSN studies have used tree-based models when transparency and low execution cost are important [6], [15]. Logistic Regression offers a compact linear decision function and can be useful when classes are approximately separable in a transformed feature space. Its lower model complexity does not guarantee the best nonlinear separation, which is reflected in the deployed dashboard’s lower displayed accuracy for Logistic Regression. Nearest-neighbor methods are also relevant because they provide a simple decision rule and a long-standing pattern-classification baseline [5]. Their storage behavior differs from that of a tree: inference can require access to many training examples, making memory and communication costs important for embedded deployment. These trade-offs explain why a classifier should be judged together with the representation and deployment location rather than in isolation.

DOI: 10.17148/IJARCCE.2026.151111

C. *Ensemble and deep-learning approaches*

Random Forest combines multiple decision trees and can capture nonlinear feature interactions with strong predictive performance [3]. The cost is a larger model and greater evaluation work than a single tree. The dashboard reflects this pattern: the 17-feature Random Forest achieves the highest visible accuracy, but it also reports a substantially larger memory figure and longer training time than the 17 -feature Decision Tree.

Recent work has explored ensemble feature selection, variable-selection ensembles, metaheuristic optimization, selective mitigation, and deep learning [8], [10], [14], [18], [20]. These methods can improve representation quality or attack coverage, but they can also introduce additional parameters, preprocessing stages, communication overhead, or model-management requirements. A constrained WSN deployment should therefore state where training occurs, where inference occurs, and which node resources are consumed by each stage.

D. *Feature selection and resource-aware design*

Feature selection reduces both the number of values transported or processed and the number of branches or parameters evaluated during prediction. The current application exposes a five-feature deployment configuration using Is CH, ADV S, SCH S, Data Sent To BS, and Expanded Energy. These variables describe node role, advertisement or scheduling behavior, traffic sent toward the base station, and an energy-related attribute in the dataset representation. Their selection does not by itself prove lower battery consumption; that claim requires measurements on the target hardware and radio stack.

IV. COMPARATIVE ANALYSIS

Table I summarizes the visible model rows in the deployed dashboard. The values are application-reported rather than independently reproduced in this paper. The dashboard indicates eight tested models; seven rows were readable in the captured comparison view and are included here. The table preserves the displayed distinction between accuracy, training time, and memory.

TABLE I
VISIBLE MODEL-COMPARISON VALUES REPORTED BY THE DEPLOYED DASHBOARD

MODEL VARIANT	ACCURACY	TRAINING TIME (S)	MEMORY (MB)
LOGISTIC REGRESSION	93.43%	116.67	67.92
DECISION TREE, 17 FEATURES	99.47%	1.96	8.55
RANDOM FOREST, 17 FEATURES	99.72%	26.17	85.93
XGBOOST, 17 FEATURES	99.67%	18.46	18.55
DECISION TREE, TOP 10	99.41%	0.99	34.30
RANDOM FOREST, TOP 10	99.59%	30.58	95.25
DECISION TREE, TOP 5	99.09%	2.07	~0^a

^aDisplayed as ~0 MB by the prototype.

The 17-feature Random Forest provides the highest visible accuracy at 99.72%. Its 0.25 percentage-point advantage over the 17-feature Decision Tree is small relative to the reported memory difference: 85.93 MB versus 8.55 MB. The Random Forest also requires 26.17 seconds of displayed training time, compared with 1.96 seconds for the 17-feature Decision Tree. The XGBoost row reaches 99.67% but still reports 18.55 MB of memory and 18.46 seconds of training time. These results show that Random Forest is the accuracy leader but not the time- and memory-efficient choice for this project.

The selected manual-deployment configuration is DT Top-5, and it is the best efficient model for the project. Its displayed accuracy is 99.09%, lower than the full-feature tree and ensembles, but the interface shows a near-zero rounded memory figure. The top-five tree also reduces the feature-ingestion burden and has a much smaller computational footprint than the Random Forest alternatives. The dashboard should not be read as proof of measured energy savings: no current-measurement trace, radio-duty-cycle measurement, node life-time experiment, or hardware benchmark is exposed in the captured application. The supported conclusion is that DT Top-5 is the preferred lightweight deployment model because it provides the required multiclass detection with lower displayed training-time and memory demands than the accuracy-leading Random Forest.

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



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151111

Logistic Regression has the lowest visible accuracy and the longest displayed training time. That result does not make linear models universally unsuitable. It indicates that, for this trained configuration and dataset, the selected linear representation did not match the class boundaries as effectively as the tree and ensemble configurations. A prediction or network lifetime. complete comparison should add precision, recall, F1 -score, confusion matrices, per-class false-negative rates, and repeated-run variation.

V. RESEARCH TRENDS AND CRITICAL GAPS

The literature shows five recurring trends. First, WSN intrusion detection has moved from fixed rules toward data-driven classifiers and hybrid protection systems [4], [9]. Second, WSN-DS has become a common benchmark, which improves comparability but risks overfitting research conclusions to one traffic generator and one feature schema [1]. Third, ensemble models and feature-selection methods are
used to improve nonlinear separation and reduce input dimensionality [8], [10]. Fourth, deep learning is increa singly evaluated for DoS detection, even when its deployment cost may exceed the capacity of individual nodes [18]. Fifth, recent studies are beginning to consider selective mitigation and edge-oriented deployment rather than detection accuracy alone [20], [22].

Several gaps remain:

1) **Hardware evidence:** Many studies report software metrics without measuring execution on the target microcontroller, radio, or gateway.

2) **Energy evidence:** Feature count and model size suggest a resource advantage, but they do not quantify energy per

^aDisplayed as ~0 MB by the prototype.

3) **Cross-dataset evaluation:** A model trained on WSN-DS may not retain its performance under a different protocol, topology, or attack generator.

4) **Class-specific reliability:** A single accuracy value can conceal high false-negative rates for a minority attack class.

5) **Adversarial robustness:** Attackers can alter rates, fields, and timing patterns to evade a detector trained on static examples.

6) **Reproducible deployment reporting:** Papers often omit preprocessing order, selected features, model serialization details, and the distinction between training time and inference latency.

VI. SYNTHESIS AND DISCUSSION

The implemented application provides a lightweight multi-class WSN DoS detection workflow. It exposes manual and bulk prediction paths and provides a model-comparison view that makes accuracy and resource trade-offs visible. The full-feature Random Forest has the highest displayed accuracy, but the five-feature Decision Tree is the selected best and efficient deployment model because it requires substantially less displayed training time and memory while retaining 99.09% displayed accuracy.

The result should be interpreted within the boundary of the available evidence. The application reports model performance on WSN-DS, but the captured interface does not reveal the train–test split, class distribution, random seed, preprocessing pipeline, calibration method, hardware platform, or inference latency. The reported training-time column cannot answer how quickly a deployed node classifies one packet. Similarly, the memory figures do not establish battery consumption. These missing details do not invalidate the application; they define the measurements required before making a hardware-deployment claim.

The implementation uses structured numerical WSN-DS features and lightweight classical machine-learning models. Its design goal is not to maximize accuracy with the largest ensemble, but to select a detector that can be trained, stored, and operated with lower resource demands. On the available dashboard evidence, DT Top-5 best matches that goal and is therefore the recommended model for the project.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151111

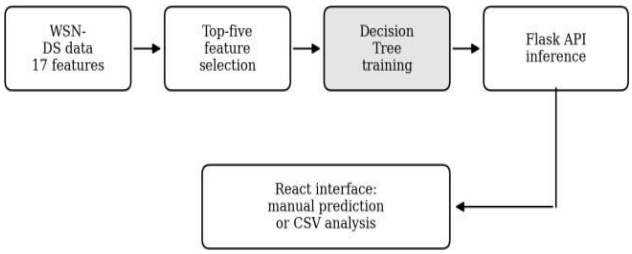


Fig. 1. Proposed lightweight deployment framework based on the implemented WSN-DS Decision Tree detector.

VII. PROPOSED LIGHTWEIGHT DETECTION FRAMEWORK

The proposed framework is a compact Decision Tree-based detector for multiclass WSN attack traffic. Model training is performed using the WSN-DS dataset, while deployment is supported through a React interface connected to a Flask prediction service. The framework accepts either the selected manual features or a CSV file containing records for bulk analysis. It returns one of five classes: Normal, Flooding, Blackhole, Grayhole, or TDMA.

The implementation uses two feature configurations. The full-feature experiments use all 17 WSN-DS input features, while the selected lightweight deployment configuration uses five features: *Is CH, ADV S, SCH S, Data Sent To BS, and Expanded Energy*. The five-feature configuration is trained and evaluated with a Decision Tree. Reducing the feature set limits the amount of data that must be collected, transferred, validated, and processed for each prediction. A shallow Decision Tree then applies a sequence of feature comparisons, making the classifier interpretable and suitable for a resource-aware implementation.

The model-selection rule is accuracy subject to lightweight constraints. The Random Forest is retained as a benchmark because it achieves the highest displayed accuracy, but it is not selected for deployment because its displayed training time and memory are substantially higher. The DT Top-5 configuration is selected as the project model because it offers the intended balance between detection capability and computational efficiency. The displayed values should be validated later through direct inference-latency, RAM, storage, processor, and energy measurements on the target platform.

Figure 1 summarizes the project workflow. The WSN-DS records provide the training data, feature selection reduces the input footprint, the Decision Tree performs classification, and the Flask API exposes predictions to the React interface. This architecture keeps the deployed detection path separate from offline model training and supports both manual and bulk analysis.

VIII. FUTURE RESEARCH DIRECTIONS

Future experiments should begin with reproducibility. The project should publish the exact train–test split, preprocessing sequence, feature-ranking method, model hyperparameters, serialization format, and software versions. The dashboard should expose per-class confusion matrices and confidence values so that a high aggregate accuracy is not mistaken for uniform class reliability.

The next stage should measure deployment behavior on representative hardware. Useful measurements include inference latency per record, peak RAM, model storage, CPU cycles, radio-on time, energy per prediction, and the effect of detection frequency on network lifetime. Measurements should be collected for both the five-feature tree and the full-feature alternatives. If the detector runs at a gateway, the study should report gateway capacity and communication overhead instead of attributing gateway savings to sensor-node energy.

Robustness evaluation should vary topology, node density, traffic rate, missing values, noisy measurements, and attack timing. Cross-dataset testing can reveal whether the detector learns attack behavior or merely recognizes artifacts of WSN-DS. A temporal split is also appropriate because random record-level splitting can place highly similar observations in both training and test sets.

The selected Decision Tree should be tested with feature-ablation studies. Removing each of the five selected features can show how node role, scheduling behavior, base-station transmission, and energy-related information affect class-wise detection. The system should also support selective mitigation only after the detector's false-positive behavior has been characterized, since an incorrect mitigation action can suppress legitimate sensing traffic.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151111

IX. CONCLUSION

This paper presented an implemented lightweight machine-learning system for multiclass DoS attack detection in WSNs. The application uses WSN-DS records, 17 features, and five output classes. Its landing page reports 374,661 records and 99.47% accuracy, while the readable model dashboard reports the highest displayed accuracy of 99.72% for a 17-feature Random Forest. The study evaluates the complete 17-feature input configuration and selects a five-feature Decision Tree for deployment. The selected model uses *Is CH, ADV S, SCH S, Data Sent To BS, and Expanded Energy*. Although Random Forest provides the highest accuracy, its larger memory demand and longer displayed training time make it unsuitable as the lightweight project choice. DT Top-5 is therefore selected as the best and most efficient deployment model for the intended WSN detector.

The current evidence supports a software-level resource-aware design, not a measured energy-saving claim. Training time is distinct from inference latency, and displayed memory is distinct from battery consumption. Hardware measurements, cross-dataset validation, per-class analysis, and adversarial testing are needed before field deployment. These measurements will determine whether the Decision Tree's software-level efficiency translates into lower energy use and longer network lifetime on the target hardware.

REFERENCES

[1] I. M. Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," *Journal of Sensors*, vol. 2016, p. 4731953, 2016.
[2] L. Alsulaiman and S. Al-Ahmadi, "Performance Evaluation of Machine Learning Techniques for DoS Detection in Wireless Sensor Network," *arXiv:2104.01963*, 2021.
[3] L. Breiman, "Random Forests," *Machine Learning, vol. 45*, no. 1, pp. 5–32, 2001.
[4] I. Butun, S. D. Morgera, and R. Sankar, "A Survey of Intrusion Detection Systems in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.
[5] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[6] M. A. Elsadig, "Detection of Denial-of-Service Attack in Wireless Sensor Networks: A Lightweight Machine Learning Approach," *IEEE Access*, vol. 11, pp. 92367–92381, 2023.
[7] H. Fares, A. Vibhute, Y. Mouniane, and H. Bouijij, "Intrusion Detection in Wireless Sensor Networks using Machine Learning," *Procedia Computer Science*, 2025.
[8] M. Fatima et al., "Towards Ensemble Feature Selection for Lightweight Intrusion Detection in Resource-Constrained IoT Devices," *Future Internet*, 2024.
[9] M. Gunasekaran and S. Periakaruppan, "A Hybrid Protection Approach for Denial of Service (DoS) Attacks in Wireless Sensor Networks," *International Journal of Electronics*, vol. 104, no. 6, pp. 993–1007, 2017.
[10] A. John, I. F. Isnin, S. H. Madni, and M. Faheem, "Cluster-Based Wireless Sensor Network Framework for Denial-of-Service Attack Detection Based on Variable Selection Ensemble Machine Learning Algorithms," *Intelligent Systems with Applications*, vol. 22, p. 200381, 2024.
[11] M. Loughmari and A. El Affar, "A Lightweight Machine Learning Approach for Denial-of-Service Attacks Detection in Wireless Sensor Networks," *International Journal of Electrical and Computer Engineering*, 2025.
[12] D. Lunkad and G. Singh, "DDoS Attack Detection Using Machine Learning for Network Performance Improvement," *International Journal of Creative Research Thoughts*, vol. 8, no. 6, 2020.
[13] S. Mends and K. S. Adu-Manu, "Evaluating Machine Learning Efficacy for DoS Intrusion Detection in Wireless Sensor Networks," 2025.
[14] A. Prasad, K. Arockiasamy, and K. Gunasekaran, "A Hybrid Metaheuristic Algorithm with Machine Learning for Detecting Denial-of-Service Attacks in Wireless Sensor Networks," *Frontiers in Artificial Intelligence*, 2026.
[15] J. R. Quinlan, C4.5: Programs for Machine Learning. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[16] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A Survey on Intrusion Detection System in IoT Networks," *Cyber Security and Applications*, vol. 3, p. 100082, 2025.
[17] M. A. Rassam, M. A. Maarof, and A. Zainal, "A Survey of Intrusion Detection Schemes in Wireless Sensor Networks," *American Journal of Applied Sciences*, vol. 9, no. 10, pp. 1636–1652, 2012.
[18] S. Salmi and L. Oughdir, "Performance Evaluation of Deep Learning Techniques for DoS Attacks Detection in Wireless Sensor Network," *Journal of Big Data*, vol. 10, no. 1, p. 17, 2023.
[19] J. Sen, "A Survey on Wireless Sensor Network Security," *arXiv:1011.1529*, 2010.
[20] H.-Y. Shim et al., "Machine Learning-Based Detection and Selective Mitigation of Denial-of-Service Attacks in Wireless Sensor Networks," *Computers, Materials & Continua*, 2025.A. Verma and V. Ranga, "Machine Learning Based Intrusion Detection Systems for IoT Applications," *Wireless Personal Communications*,



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151111

2019.

[21] R. Walee, A. Dey, and A. Das, "A Lightweight Ensemble Model for IoT Intrusion Detection on Resource-Constrained Edge Devices," in *Proc. IEEE WIECON-ECE*, 2025.

[22] K. Yasotha and K. Meenakshisundaram, "Machine Learning-Based Intrusion Detection for Mitigating Denial of Service Attacks in Wireless Sensor Networks," in *Proc. Int. Conf. Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2023.