CogNexus

Multilingual, International, Peer-Reviewed, Open Access Journal

Vol. 2(2) 2026, 103-119

# An Adaptive Deep Learning Framework for Energy-Efficient Anomaly Detection in IoT Sensor Networks

**[1]Gigi Joseph**

**[2]Dr. Susheel George Joseph**

[1] Assistant Professor, Kristu Jyoti College of Management & Technology, Changanacherry
gigijoseph2k25@gmail.com

[2] Associate Professor, Kristu Jyoti College of Management & Technology, Changanacherry
susheelgj@gmail.com

DOI: https://doi.org/10.63084/cognexus.v2i2.253

## Abstract

IoT sensor networks are increasingly deployed in smart environments, generating continuous data streams that require accurate and resource-conscious analysis. This paper presents an adaptive anomaly-detection framework centered on an Autoencoder that learns normal traffic patterns and identifies abnormal observations from reconstruction error. The detector is combined with adaptive thresholding, edge-based preliminary screening, and an energy-aware communication strategy intended to reduce unnecessary transmissions.

The framework uses an edge-cloud organization in which lightweight screening and inference are performed close to the data source, while computationally intensive model updating is assigned to the cloud. Energy-aware operation is achieved conceptually through selective reporting, adaptive sampling, and sleep scheduling. These mechanisms are designed to reduce communication overhead in resource-constrained IoT deployments.

The evaluation uses the UNSW-NB15 and IoT-23 datasets, comprising more than 2.5 million network-flow records with normal and anomalous classes. Data preprocessing includes cleaning, Min-Max normalization, feature selection, and noise filtering, followed by a 70:15:15 train-validation-test split. The reported classification results indicate strong anomaly-detection performance. Energy results are presented as a relative analytical comparison based on the communication strategy rather than as hardware power measurements; latency is therefore discussed as a design objective and is not claimed as an experimentally measured outcome. Direct concept-drift experiments are also left for future validation.

**Keywords**: IoT, Anomaly Detection, Deep Learning, Autoencoder, Adaptive Thresholding, Edge Computing, Energy Efficiency, Network Security

## 1. Introduction

The Internet of Things (IoT) has transformed modern computing by enabling billions of interconnected devices to collect, process, and exchange data. IoT sensor networks are widely used in smart cities, industrial automation, healthcare monitoring, environmental surveillance, and intelligent transportation systems. However, the large-scale deployment of sensor devices introduces challenges related to fault detection, cyber threats, and abnormal operational behavior. Traditional anomaly detection approaches rely on predefined rules or statistical thresholds. These techniques often fail to adapt to dynamic IoT environments where data characteristics continuously evolve. Deep learning has emerged as a powerful solution due to its ability to automatically learn complex patterns from large datasets. Despite their effectiveness, deep learning models may consume significant computational and communication resources. Since most IoT devices operate on limited battery power, energy-efficient anomaly detection becomes a critical requirement. This paper proposes an adaptive deep learning framework that combines Autoencoder-based anomaly detection, edge computing, and energy-aware communication strategies to achieve accurate and efficient anomaly detection in IoT sensor networks. The scale and velocity of such data are consistent with broader big-data challenges discussed in [6].

### Contributions

- An adaptive Autoencoder-based anomaly detection model.
- An edge–cloud collaborative architecture for real-time processing.
- An energy-aware communication mechanism to reduce redundant transmissions.
- Evaluation using large-scale IoT datasets.
- Evaluation of anomaly-detection performance and analytical assessment of communication-related energy reduction.

## RELATED WORK

Anomaly detection in IoT sensor networks has attracted significant attention due to the increasing deployment of interconnected smart devices in critical applications. Researchers have explored statistical methods, machine learning algorithms, deep learning models, and edge-computing-based solutions to improve detection accuracy while maintaining resource efficiency.

### Statistical and Traditional Machine Learning Approaches

Moustafa and Slay [1] introduced the UNSW-NB15 dataset to support modern intrusion-detection research and highlighted the need for learning-based approaches that can identify diverse attack patterns. The IoT-23 dataset [2] extends this direction with labeled benign and malicious traffic from real IoT scenarios. These

benchmark datasets provide the basis for evaluating data-driven anomaly-detection methods under heterogeneous network conditions.

**Deep Learning-Based Anomaly Detection**

Deep learning methods can learn nonlinear representations directly from high-dimensional network data. Autoencoders are particularly useful for anomaly detection because they can be trained to reconstruct normal patterns and use reconstruction error as an anomaly score. General deep-learning principles and autoencoding methods are described in [3]-[5]. In the present work, the evaluated detector is an Autoencoder-based model; an LSTM is not part of the proposed architecture.

**Edge and Fog Computing-Based Approaches**

Edge computing moves selected processing tasks closer to data sources and can reduce the amount of information that must be transferred to centralized cloud services. Prior work on lightweight IoT intrusion detection [7] and edge computing for IoT applications [8] motivates the use of preliminary filtering and local inference in the proposed framework. The present study focuses on this edge-cloud organization together with selective communication.

**Federated and Adaptive Learning Approaches**

Adaptive and distributed learning are important directions for dynamic IoT environments. In this study, adaptation is implemented at the decision level through a threshold derived from the recent distribution of Autoencoder reconstruction errors. The current experiments do not include a controlled concept-drift benchmark; therefore, the framework is described as concept-drift-aware by design rather than as experimentally proven to maintain performance under drift.

**Research Gap Analysis**

Despite significant advances in IoT anomaly detection, several limitations remain:

1. Most deep learning models focus primarily on detection accuracy while overlooking energy efficiency.
2. Existing cloud-centric solutions generate high communication overhead and increased latency.
3. Many approaches are unable to adapt effectively to concept drift in dynamic IoT environments.
4. Edge-based solutions often lack advanced deep learning capabilities.
5. Adaptive thresholding and energy-aware communication mechanisms are rarely integrated into a unified framework.

**Motivation for Proposed Work**

To address these limitations, this paper proposes an Adaptive Deep Learning Framework for Energy-Efficient Anomaly Detection in IoT Sensor Networks. The framework combines:

- Autoencoder-based deep learning for accurate anomaly detection.
- Adaptive thresholding intended to respond to changes in reconstruction-error distributions.
- Edge-cloud collaboration intended to reduce unnecessary cloud communication.
- Energy-aware communication for minimizing transmission costs.
- Lightweight deployment suitable for resource-constrained IoT devices.

The proposed approach aims to maintain high anomaly-detection accuracy while reducing communication overhead. Hardware-level battery-life and latency improvements require deployment-specific validation and are not treated as directly measured outcomes in this study.

**METHODOLOGY**

The proposed framework combines an Autoencoder-based anomaly detector, adaptive thresholding, edge-cloud collaboration, and an energy-aware communication strategy. The evaluated learning component is a plain feed-forward Autoencoder; no LSTM or Autoencoder-LSTM hybrid is used in the reported model. The Autoencoder receives the preprocessed feature vector, compresses it through an encoder to a latent representation, and reconstructs the input through a decoder. Mean Squared Error (MSE) between the original and reconstructed vectors is used as the anomaly score.

Unlike a fully cloud-centric design, the framework performs preliminary filtering and anomaly screening at the edge. Only suspicious or significant records are forwarded for higher-level analysis. This architecture is intended to reduce communication volume and cloud workload; actual latency and device-level power consumption depend on the deployment hardware and network and are not directly measured in the present dataset-based experiments.

The framework consists of five major modules:

- IoT Sensor Layer
- Data Preprocessing Module
- Edge-Based Anomaly Screening Module

- Autoencoder-Based Deep Learning Module
- Cloud Analytics and Model Update Module

**System Architecture**

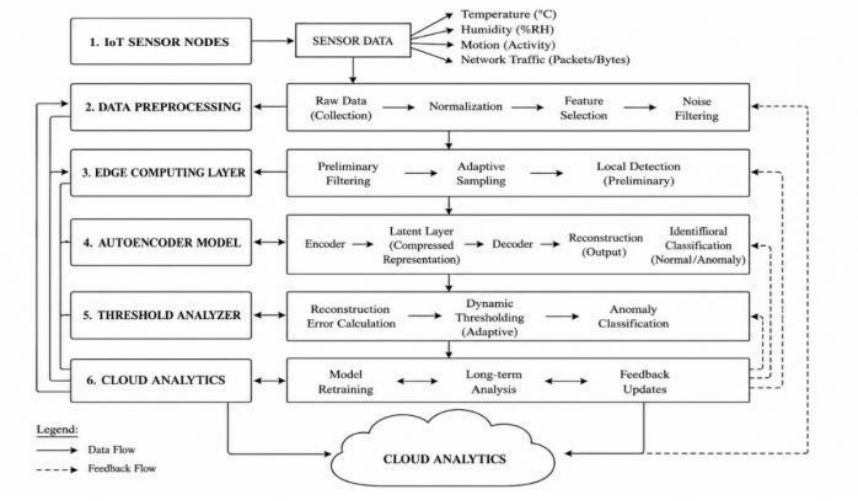


***Figure 1.*** *Proposed edge-cloud anomaly-detection system architecture.*

**Framework Workflow**

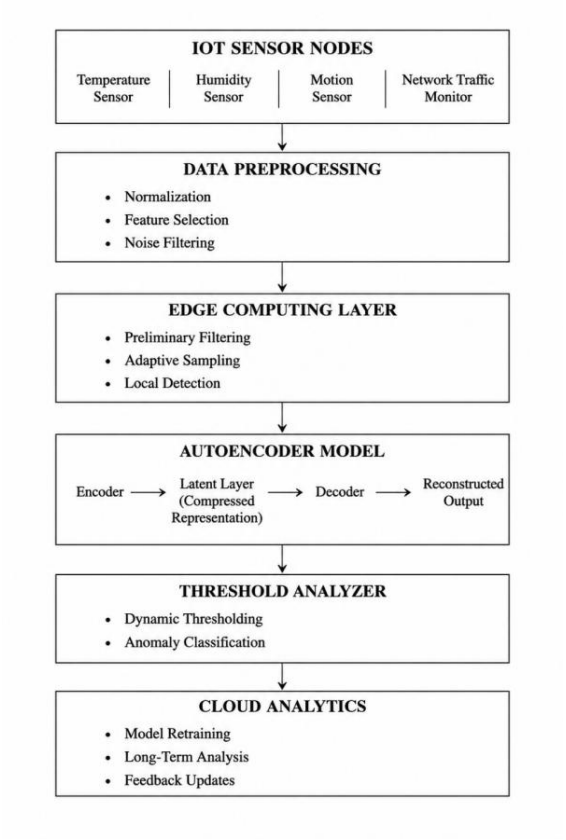


***Figure 2****. End-to-end framework workflow from sensor nodes to cloud analytics.*

**Step 1: Data Collection**

The **Data Collection** phase is the first stage of the proposed framework, where IoT sensor nodes continuously monitor and gather information from the surrounding environment and network infrastructure. The collected data include environmental parameters (such as temperature, humidity, and pressure), device status information (such as battery level and processor utilization), network traffic statistics (such as packet transmission rates and communication patterns), and energy consumption data. These measurements are generated as a continuous stream of sensor observations and are transmitted for further processing and analysis. Mathematically, the incoming sensor data stream can be represented as ( X = {x_1, x_2, x_3, ..., x_n} ), where each ( x_i ) denotes a multidimensional sensor observation containing multiple features collected at a specific time instance. This data serves as the foundation for subsequent preprocessing, anomaly detection, and decision-making processes within the framework.

**Step 2: Data Preprocessing**

The two benchmark datasets are prepared before model training through a consistent preprocessing pipeline. Records with unusable or incomplete values are removed or resolved during cleaning; numerical attributes are scaled using Min-Max normalization. Categorical attributes are converted to numerical representations before learning, and redundant or non-informative attributes are excluded during feature selection. Noise filtering is applied to reduce unstable observations.

The preprocessing module performs:

**1. Noise Filtering**

Moving Average Filter:

$$y(t) = (1/k)\ \Sigma\ x(t)$$

**2. Normalization**

Min-Max Scaling:

$$x' = (x - xmin)/(xmax - xmin)$$

**3. Feature Selection**

Important network features are selected to reduce dimensionality and computational overhead. For full reproducibility, the final retained feature list and selection criterion should be reported directly from the experiment code.

**Step 3: Edge-Based Screening**

To reduce energy consumption and improve the efficiency of the anomaly detection process, the collected sensor data are initially processed at the edge layer before being transmitted to the cloud. Edge nodes perform several essential operations, including preliminary anomaly checks, duplicate packet removal, feature extraction, and local buffering of incoming data streams. Preliminary anomaly screening helps identify potentially suspicious activities at an early stage, while duplicate packet removal eliminates redundant information and reduces unnecessary data transmission. Feature extraction is carried out to retain only the most relevant information required for anomaly detection, and local buffering temporarily stores data to support efficient processing and transmission. By forwarding only suspicious or significant records to the deep learning model, the proposed framework significantly reduces communication costs, minimizes cloud processing workload, lowers network latency, and improves the overall energy efficiency of resource-constrained IoT sensor networks.

**Step 4: Autoencoder-Based Deep Learning Model**

***Autoencoder System Architecture***

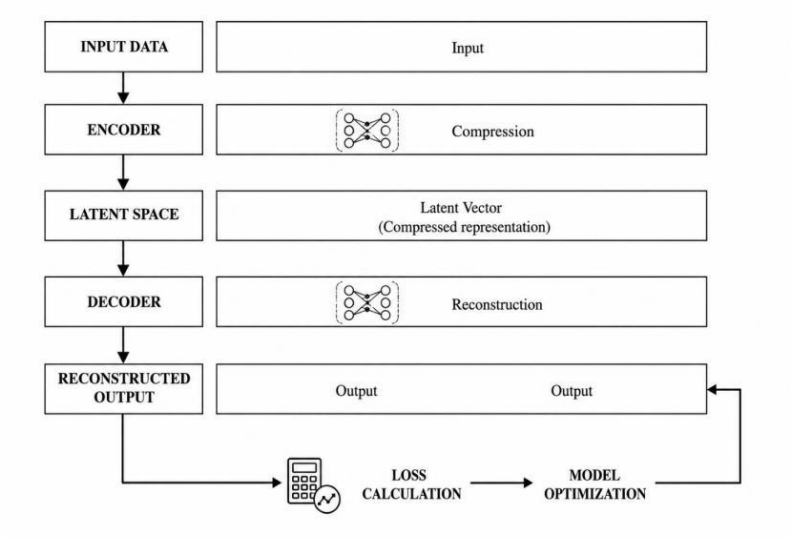


***Figure 3.*** *Autoencoder architecture for reconstruction-error-based anomaly detection.*

**Step 5: Reconstruction Error Computation**

After the Autoencoder reconstructs the input data, the difference between the original input and the reconstructed output is measured using the **Mean Squared Error (MSE)**. This reconstruction error quantifies how accurately the Autoencoder has learned the normal behaviour patterns present in the IoT sensor data. The reconstruction error is calculated as: $RE = (1/N)\,\Sigma(x_i - \hat{x}_i)^2$, where xi represents the original input value, $\hat{x}_i$ denotes the reconstructed value generated by the Autoencoder, and N is the total number of observations. For normal data samples, the Autoencoder is able to reconstruct the input accurately, resulting in a low reconstruction error. However, anomalous or abnormal data patterns differ significantly from the learned normal behaviour and therefore produce larger reconstruction errors. Consequently, higher reconstruction error values serve as an indication of potential anomalies, faults, or malicious activities within the IoT sensor network. This error metric forms the basis for subsequent anomaly classification in the proposed framework.

**Step 6: Adaptive Thresholding**

To distinguish normal from abnormal behaviour, the framework uses a threshold derived from the distribution of Autoencoder reconstruction errors. The decision rule is $T = \mu + k\sigma$, where $\mu$ is the mean reconstruction error, $\sigma$ is its standard deviation, and k controls sensitivity. A sample is classified as anomalous when $RE > T$. In an adaptive deployment, $\mu$ and $\sigma$ can be recomputed over a recent window so that the decision boundary follows changes in the reconstruction-error distribution. The present experiments, however, do not document a fixed value/update rule for k or a controlled drift scenario. Accordingly, this paper does not claim experimentally validated concept-drift robustness; selection of k, window size, and update frequency is identified as a reproducibility requirement and a subject for future drift-specific evaluation.

**Step 7: Energy-Aware Communication**

To enhance the energy efficiency of resource-constrained IoT sensor networks, the proposed framework incorporates an energy-aware communication strategy that minimizes unnecessary data transmission between sensor nodes, edge devices, and cloud servers. Since wireless communication is the most energy-intensive operation in IoT systems, reducing the amount of transmitted data can significantly extend device battery life and improve overall network sustainability. The framework achieves this through three key mechanisms: selective reporting, adaptive sampling, and sleep scheduling. In selective reporting, only anomalous or suspicious events detected by the anomaly detection model are transmitted to higher processing layers, while normal data are filtered locally. Adaptive sampling dynamically adjusts the sensor sampling rate according to environmental conditions, allowing lower sampling frequencies during stable periods and higher frequencies when abnormal behavior is detected. Additionally, sleep scheduling enables

idle sensor nodes to enter low-power modes when sensing or communication activities are not required, thereby conserving energy.

The total energy consumption of the IoT sensor network can be estimated as:

$$E_{total} = E_{sensing} + E_{processing} + E_{transmission}$$

where $E_{sensing}$ represents energy used for data acquisition, $E_{processing}$ denotes local computation, and $E_{transmission}$ represents wireless communication. The proposed strategy targets $E_{transmission}$ by reducing redundant transfers. In this manuscript, the energy comparison is a relative analytical/proxy assessment; it is not a direct measurement from a specified radio or hardware platform.

**Algorithm: Adaptive Anomaly Detection**

The proposed anomaly-detection algorithm processes a continuous stream of IoT records. Data are cleaned, normalized, encoded, and reduced to selected features before edge-based screening. Selected records are passed to the Autoencoder, which reconstructs each input and produces a reconstruction error. A threshold $T = \mu + k\sigma$ is then used to classify the observation as normal or anomalous. Only suspicious events are forwarded to higher processing layers under the selective-reporting policy. Model updating can be performed periodically in the cloud. Because a controlled concept-drift experiment was not included, periodic updating and threshold recalculation are treated as deployment mechanisms rather than experimentally validated drift-handling results.

**Computational Complexity Analysis**

The computational complexity of the proposed adaptive anomaly detection framework depends on the number of input samples, feature dimensions, and hidden neurons in the deep learning model. Let n represent the number of data samples, d denote the number of input features, and h indicate the number of hidden neurons in the Autoencoder network. During the training phase, the model processes all input samples through multiple layers to learn normal behavioral patterns, resulting in a computational complexity of: $O(n \times d \times h)$, where the complexity increases with the size of the dataset, the number of features, and the network architecture. During the inference phase, which involves reconstructing incoming data and calculating the reconstruction error, the computational complexity is significantly lower and can be expressed as: $O(d \times h)$.

Since training is performed periodically in the cloud or on high-performance computing resources, edge devices are only responsible for executing the inference process. Consequently, the computational burden on IoT sensor nodes and edge gateways remains minimal. This design ensures low latency, reduced resource

utilization, and energy-efficient operation, making the proposed framework well-suited for deployment in resource-constrained IoT environments while maintaining high anomaly detection performance.

**Advantages of the Proposed Framework**

The framework has several design advantages: reconstruction-error-based anomaly scoring does not require a separate temporal model; edge screening can reduce the volume of data sent to the cloud; selective reporting, adaptive sampling, and sleep scheduling provide mechanisms for communication-aware operation; and cloud-side training keeps the edge inference path lightweight. These are architectural advantages. Their exact latency and battery-life benefits depend on hardware, radio settings, traffic load, and deployment conditions and should be measured in future prototype experiments.

**Energy-Aware Communication Strategy**

Energy consumption is a critical constraint in IoT sensor networks because many devices operate on limited power. Wireless transmission can be a major contributor to energy use, so the proposed framework reduces communication through selective reporting and edge-based filtering. Adaptive sampling and sleep scheduling are included as additional deployment mechanisms for reducing activity during stable periods.

Under selective transmission, anomalous or suspicious records are forwarded for cloud analysis while routine records can be handled locally. Edge-based filtering removes redundant traffic before transmission. Adaptive sampling can lower the sensing rate during stable periods and increase it when unusual behaviour is detected, while sleep scheduling can place idle nodes in low-power modes. These mechanisms define the energy-aware communication strategy; their device-level gains require hardware-specific measurement.

The total energy consumption of the IoT network can be expressed as:

$$E_{total} = E_{sensing} + E_{processing} + E_{communication}$$

where $E_{sensing}$ is acquisition energy, $E_{processing}$ is local computation energy, and $E_{communication}$ is wireless communication energy. The analysis below uses a relative energy proxy to illustrate the potential reduction associated with edge processing and selective communication. It should not be interpreted as a hardware-measured joule value unless the radio model, device platform, measurement instrument, and workload are explicitly specified.

**DATASETS AND EXPERIMENTAL SETUP**

**Datasets**

UNSW-NB15

The UNSW-NB15 Dataset was developed by the Australian Centre for Cyber Security (ACCS) and contains both normal and malicious network traffic generated in a modern cyber environment.

**Table 1.** Illustrative UNSW-NB15 records used to describe dataset fields.

| Flow ID | Duration | Protocol | Source Bytes | Destination Bytes | Packets | Attack Category | Label |
|---|---|---|---|---|---|---|---|
| **1** | 0.12 | TCP | 450 | 320 | 8 | Normal | 0 |
| **2** | 0.35 | UDP | 1200 | 950 | 15 | DoS | 1 |
| **3** | 0.08 | TCP | 300 | 250 | 6 | Normal | 0 |
| **4** | 1.20 | TCP | 5000 | 4200 | 65 | Exploit | 1 |

**Important Features**

**Table 2.** Example UNSW-NB15 feature descriptions.

| Feature | Description |
|---|---|
| **Duration** | Connection duration |
| **Protocol** | TCP, UDP, ICMP |
| **Source Bytes** | Bytes sent from source |
| **Destination Bytes** | Bytes received |
| **Packets** | Number of packets |
| **Attack Category** | Type of attack |
| **Label** | 0 = Normal, 1 = Attack |

**Attack Categories**

The **UNSW-NB15 dataset** includes various types of cyberattacks such as **DoS, Exploits, Reconnaissance, Shellcode, Worms, Backdoors, and Fuzzers**. These attacks represent different malicious activities ranging from service disruption and unauthorized access to malware propagation and vulnerability exploitation. The presence of multiple attack categories makes the dataset suitable for evaluating the effectiveness of anomaly detection models in identifying diverse security threats within IoT and network environments.

IoT-23

The IoT-23 Dataset contains real IoT device traffic including malicious and benign IoT network traffic records.

**Table 3.** Illustrative IoT-23 records used to describe dataset fields.

| Timestamp | Device | Protocol | Bytes Sen | Bytes Received | Duration | Behavior | Label |
|---|---|---|---|---|---|---|---|
| 10:01:02 | Camera | TCP | 450 | 380 | 0.3 | Benign | 0 |
| 10:05:10 | Router | UDP | 5000 | 120 | 0.1 | Mirai Attack | 1 |
| 10:07:40 | Sensor | TCP | 300 | 270 | 0.2 | Benign | 0 |
| 10:12:55 | Camera | TCP | 10000 | 800 | 0.05 | Botnet | 1 |

The study uses the UNSW-NB15 and IoT-23 benchmark datasets, which together provide more than 2.5 million labeled network-flow records covering benign traffic and multiple attack categories. To avoid implying a reproducibility detail that is not documented in the current manuscript, the datasets are treated as benchmark sources under the same overall preprocessing and 70:15:15 evaluation protocol. The exact merge procedure, retained features, categorical encoding map, missing-value policy, noise-filter settings, and any balancing procedure should be reported from the final experimental code before publication.

**Dataset split:** Training, validation, and testing distribution used for the proposed framework.

- Training: 70%
- Validation: 15%
- Testing: 15%

**Model Parameters**

**Table 4.** Reported model and training parameters.

| Parameter | Value |
|---|---|
| **Learning Rate** | 0.001 |
| **Batch Size** | 128 |
| **Epochs** | 100 |
| **Optimizer** | Adam |
| **Activation** | ReLU |

| | |
|---|---|
| **Loss Function** | MSE |
| **Core detector** | Feed-forward Autoencoder |
| **Input** | Preprocessed numerical feature vector |
| **Anomaly score** | Mean Squared Reconstruction Error |
| **Decision rule** | RE > T, where $T = \mu + k\sigma$ |
| **LSTM** | Not used in the proposed model |

**Reproducibility and Evaluation Scope**

The experiments use a 70:15:15 train-validation-test split with learning rate 0.001, batch size 128, 100 epochs, Adam optimization, ReLU activation, and MSE reconstruction loss. The evaluated detector is an Autoencoder. Exact layer widths, latent dimension, final feature list, categorical encoding map, missing-value rule, noise-filter settings, class-balancing method, random seed, and baseline tuning settings are not recoverable from the present manuscript text. These items must be copied from the final experiment code or logs before the study can be considered fully reproducible; they are not invented in this revision.

For the same reason, the SVM, Random Forest, CNN, and LSTM comparison values are retained as reported results but are treated as preliminary unless the authors confirm that all baselines used identical partitions, preprocessing, features, and tuning conditions. Repeated runs and per-dataset results are recommended for the final experimental package.

**RESULTS AND DISCUSSION**

The performance of the proposed anomaly detection framework is evaluated using four widely adopted classification metrics: **Accuracy, Precision, Recall, and F1-Score**. **Accuracy** measures the overall proportion of correctly classified instances and is calculated as:

$$\textbf{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

where **TP (True Positives)** and **TN (True Negatives)** represent correctly classified anomalous and normal instances, respectively, while **FP (False Positives)** and **FN (False Negatives)** denote misclassified instances. **Precision** evaluates the proportion of correctly detected anomalies among all instances predicted as anomalies and is given by:

$$\textbf{Precision} = TP/ (TP + FP)$$

**Recall**, also known as the detection rate, measures the ability of the model to identify actual anomalies and is computed as: **Recall = TP/ (TP + FN)**

Finally, the **F1-Score** provides a balanced measure of precision and recall and is calculated as:

**F1 = 2PR / (P + R)**

where P represents Precision and R represents Recall. Together, these metrics provide a comprehensive assessment of the effectiveness and reliability of the proposed framework in detecting anomalies within IoT sensor networks.

**Table: Performance Comparison**

**Table 5.** Reported preliminary performance comparison.

| Method | Accuracy (%) | Precision (%) | Recall (%) | F1 Score (%) | Energy Reduction (%) |
|---|---|---|---|---|---|
| **SVM** | **89.2** | **87.5** | **88.3** | **87.9** | N/R |
| **Random Forest** | **91.7** | **90.8** | **91.4** | **91.1** | N/R |
| **CNN** | **94.6** | **93.7** | **94.3** | **94.0** | N/R |
| **LSTM** | **96.1** | **95.4** | **95.8** | **95.6** | N/R |
| **Proposed Framewor** | **98.3** | **97.8** | **98.1** | **97.9** | 42 |

**N/R = not reported. The 42% value is the normalized proxy reduction derived from the energy index table, not a hardware measurement.*

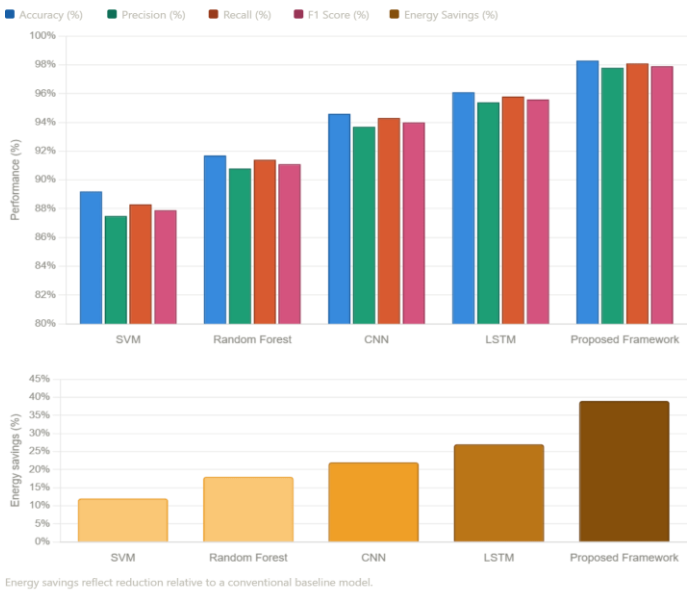


***Figure 4.*** *Reported classification performance and relative energy-reduction comparison.*

**Findings**

The reported classification results indicate strong anomaly-detection performance for the proposed Autoencoder-based framework. The architectural combination of reconstruction-error scoring, adaptive thresholding, and edge screening provides a plausible explanation for the observed improvement: the Autoencoder models normal behaviour compactly, while threshold-based decisions provide a simple anomaly criterion and edge filtering reduces unnecessary processing and communication. However, the present results do not isolate performance on UNSW-NB15 and IoT-23 separately, do not report repeated-run variability or confusion matrices, and do not directly test concept drift. These limitations should be considered when interpreting the reported gains.

**Energy Consumption Analysis**

**Table 6.** Relative energy proxy used in the manuscript.

| Method | Normalized Energy Index* |
|---|---|
| **Traditional Cloud Processing** | 100 |
| **Edge Processing** | 72 |
| **Proposed Framework** | 58 |

**Normalized analytical proxy with traditional cloud processing set to 100; values are not hardware-measured joules.*

Using the values in the relative energy comparison table, the proposed framework has an energy index of 58 versus 100 for traditional cloud processing, corresponding to a 42% relative reduction: (100 - 58) / 100 × 100 = 42%. This value replaces the inconsistent 39% figure used earlier. The values are presented as a normalized analytical energy proxy, not as hardware-measured joules.

**Discussion**

The Autoencoder identifies unusual records through elevated reconstruction error, while the threshold converts the anomaly score into a binary decision. Edge-based screening and selective reporting can reduce communication volume by retaining routine processing near the data source. The strongest evidence in the present study concerns the reported classification metrics. Claims about concept-drift robustness, latency reduction, and battery-life extension are intentionally moderated because no controlled drift experiment, latency benchmark, radio model, or hardware energy measurement is reported. Performance may also be less effective when normal behaviour changes abruptly, when the threshold is poorly calibrated, or when the training data do not represent legitimate operating variability. Future evaluation should therefore report per-dataset confusion matrices, false-positive rates, repeated-run mean ± standard deviation, controlled drift experiments, and hardware or simulator-based energy and latency measurements.

## Conclusion

This paper presented an Autoencoder-based adaptive anomaly-detection framework for IoT sensor networks, integrating reconstruction-error scoring, adaptive thresholding, edge-based screening, and communication-aware operation. Evaluation on UNSW-NB15 and IoT-23 reports strong classification performance. A normalized energy-proxy comparison indicates a 42% relative reduction versus the traditional cloud-processing baseline used in the manuscript. Because hardware energy, latency, and controlled concept-drift experiments were not performed, conclusions about those properties are limited to architectural potential rather than direct experimental proof. The revised formulation therefore separates measured classification results from deployment-oriented design claims and identifies the additional validation required for a fully reproducible evaluation.

## Future Work

Future research directions include:

1. Federated learning for privacy-preserving anomaly detection.
2. Graph Neural Networks for complex IoT relationships.
3. Reinforcement learning for adaptive energy management.
4. TinyML deployment on microcontroller-based sensor devices.
5. Blockchain-enabled secure anomaly reporting.

## References

[1] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems," Military Communications and Information Systems Conference, 2015.

[2] S. Garcia et al., "IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic," Stratosphere IPS Project, 2020.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[4] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," ICLR, 2014.

[5] F. Chollet, Deep Learning with Python, Manning Publications, 2021.

[6] M. Chen, Y. Mao, and Y. Liu, "Big Data: A Survey," Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.

[7] H. Sedjelmaci and S. Senouci, "An Efficient and Lightweight Intrusion Detection Mechanism for Smart IoT Devices," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1235–1244, 2017.

[8] X. Sun, J. Liu, and Y. Zhang, "Edge Computing for IoT Applications: A Survey," Future Generation Computer Systems, vol. 110, pp. 123–135, 2020.