International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/08 Volume 12, Issue 1, January – February (2025)

RESEARCH ARTICLE

Hybrid Dynamic Kernel Neural Learning for Efficient Anomaly Detection in Wireless Sensor Networks

T. Selvakumar
Department of Computer and Information Science, Annamalai University, Annamalai Nagar, Tamil Nadu, India.
selva5709@gmail.com

M. Jeyakarthic
Department of Computer and Information Science, Annamalai University, Annamalai Nagar, Tamil Nadu, India.
✉ jeya_karthic@yahoo.com

Received: 28 November 2024 / Revised: 14 January 2025 / Accepted: 28 January 2025 / Published: 28 February 2025

Abstract – Wireless Sensor Networks (WSNs) are increasingly used for real-time monitoring across various critical applications, including environmental sensing, infrastructure monitoring, and healthcare. However, WSNs face significant challenges in anomaly detection due to their resource constraints, dynamic topology, and the complexity of high-dimensional sensor data with varying patterns. These challenges make traditional methods ineffective, highlighting the need for innovative approaches. Traditional anomaly detection methods often struggle to handle complex, high-dimensional sensor data with varying patterns. To address these challenges, we propose Hybrid Dynamic Kernel Neural Learning (HDK-NL), a novel framework that integrates deep neural networks with dynamic kernel selection for efficient and accurate anomaly detection in WSNs. HDK-NL ensuring robust detection of both spatial and temporal anomalies. A dynamic kernel hierarchy is introduced, which automatically selects kernel types (Gaussian, polynomial, linear) based on statistical properties of the extracted features, improving the algorithm's capacity to discern intricate patterns. The algorithm employs a multi-scale scoring system that aggregates anomaly scores from multiple layers, considering both local and global contexts. To optimize energy consumption in WSNs, context-aware adaptive thresholding is used to minimize false positives and reduce unnecessary transmissions. The proposed method is evaluated on real-world sensor data, demonstrating improved detection accuracy, reduced false alarms, and significant energy savings. HDK-NL offers a scalable and adaptive solution for anomaly detection in WSNs, making it suitable for resource-constrained environments that require real-time processing.

Index Terms – Wireless Sensor Networks, Anomaly Detection, Hybrid Dynamic Kernel Learning, Deep Neural Networks, Convolutional Neural Networks, Long Short-Term Memory, Dynamic Kernel Selection.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are extensively used in a several of real-time applications, such as environmental monitoring, healthcare, industrial systems, and smart cities. WSNs consist of distributed sensor nodes that continuously collect data, which is then transmitted to a central node or processing unit. Given the critical nature of these applications, it is crucial to ensure that the sensor network operates reliably and efficiently [1]. However, WSNs are often prone to various challenges, including sensor malfunctions, network failures, environmental disturbances, and malicious attacks. These issues may lead to abnormal or anomalous data patterns that compromise the system's performance and data integrity [2].

The identification of anomalies in WSNs involves recognizing points of data or events that markedly diverge from typical behavior [3]. This can include detecting sensor faults, environmental changes, network intrusions, or any irregularity that affects the accuracy and reliability of the data being transmitted. Anomaly detection plays a critical role in the maintenance and operation of WSNs, as it allows for early identification of issues that may escalate into larger problems, thereby reducing downtime and ensuring that system reliability is maintained [4].

Nonetheless, anomaly identification in wireless sensor networks is especially difficult owing to the unique features of these systems. First, sensor nodes often have limited computational resources, memory, and energy, making it difficult to implement complex detection models. Second, sensor data is typically high-dimensional, noisy, and unstructured, complicating the identification of anomalies [5]. Moreover, the network environment is dynamic, with sensor

RESEARCH ARTICLE

nodes frequently being added or removed, leading to fluctuations in the data distribution. Traditional rule-based systems often fail to adapt to the evolving nature of WSNs and may not be effective at detecting subtle anomalies or complex patterns in the data [6].

In recent years, machine learning (ML) and deep learning (DL) methods have emerged as promising solutions for anomaly detection, offering superior accuracy and adaptability [7]. These approaches can autonomously discern intricate trends in information and adjust to environmental changes. Nonetheless, despite their capabilities, current deep learning algorithms often encounter difficulties when used in WSNs [8]. Many deep learning methods require large amounts of labeled data for training, which is often unavailable in real-world WSN scenarios. Additionally, the high computational cost and memory requirements of deep learning models may not be suitable for the resource-constrained nature of sensor nodes [9][10].

A novel approach, Hybrid Dynamic Kernel Neural Learning (HDK-NL), which combines deep neural networks with dynamic kernel methods for efficient and accurate anomaly detection in WSNs. Our approach combines Convolutional Neural Networks (CNNs) as well as Long Short-Term Memory (LSTM) in the sensor data. By using CNNs, we can effectively capture local spatial patterns within the data, while LSTMs help model long-term temporal dependencies, ensuring that both types of anomalies—spatial and temporal—are detected. In addition, the model integrates a dynamic kernel hierarchy, which adapts the kernel functions.

The core idea behind HDK-NL is to dynamically adjust the kernel type based on the feature distributions at each layer of the network. This dynamic kernel selection allows the model to capture a wide variety of patterns in the data, from simple linear relationships to more complex, non-linear interactions. The hierarchical structure of the model enables multi-scale anomaly detection, which are more comprehensive understanding of the data. Furthermore, the algorithm incorporates context-aware adaptive thresholding and environmental context. This ensures that the model remains flexible and can handle varying levels of anomaly severity across different operating conditions.

One of the key advantages of HDK-NL is its energy-efficient design. WSNs are typically deployed in environments with limited energy resources, and frequent data transmission for anomaly reporting can quickly drain the battery of sensor nodes. By employing context-aware anomaly scoring and adaptive thresholding, HDK-NL reduces the number of false positives and unnecessary anomaly alerts, minimizing data transmission and conserving energy. This makes the model particularly well-suited for deployment in large-scale, resource-constrained WSNs where energy efficiency is a top priority.

As WSNs become more prevalent in mission-critical applications, ensuring the reliability of these networks becomes increasingly important. Anomalies can arise due to various factors, such as sensor failures, environmental disturbances, or even malicious attacks on the network. Traditional methods for anomaly detection, while useful in some contexts, struggle to keep up with the evolving data patterns found in real-world WSNs. These methods may not effectively handle high-dimensional data or adapt to the changing conditions of the network. Moreover, many traditional techniques are not well-suited for deployment in resource-constrained environments where sensor nodes have limited computational power and energy. The main objectives of this work are:

- Develop an Adaptive Anomaly Detection Model: By incorporating dynamic kernel learning and deep neural networks, the model will be able to adapt to different types of data distributions and detect a wide range of anomalies, including both spatial and temporal irregularities.
- Enhance Energy Efficiency: The proposed model will minimize unnecessary transmissions and energy consumption by using adaptive thresholds and focusing only on significant anomalies.
- Improve Detection Accuracy: The hybrid model will leverage both spatial and temporal feature extraction to identify anomalies with greater precision, reducing false positives and improving the model's overall effectiveness.
- Enable Real-Time Anomaly Detection: The algorithm will be optimized for real-time processing, allowing it to detect anomalies as they occur and trigger timely alerts for network maintenance or intervention.

The paper is structured as follows: Section 2 presents the state-of-the-art in anomaly detection for WSNs, identifying gaps and limitations in current research. Section 3 proposes the Hybrid Dynamic Kernel Neural Learning (HDK-NL) framework, which consists of its architecture, its major components such as the dynamic kernel hierarchy and multi-scale scoring. Section 4 presents the experimental findings and includes a comparative analysis to highlight the efficacy of the proposed method. Section 5 closes this paper and discusses directions for further research into anomaly detection on WSNs.

2. RELATED WORKS

Wavelet_Kernel_Network with Omni-Scale-Convolution (WKN-OC) model is designed for detecting anomaly in Intelligent Transportation Systems (ITS) [11]. It adaptively selects optimal scales, emphasizes high-frequency signals, and extracts valuable features for improved anomaly

RESEARCH ARTICLE

detection. The model is validated on the SPMD dataset, achieving high accuracy in detecting mixed and multi-anomaly scenarios. The computational complexity of Omni-Scale Convolution might increase the processing time.

Enhanced Transient Extreme Learning method detects anomalies in WSN data through three stages: data compression using Piecewise Aggregate Approximation, prediction via Extreme Learning Machine (ELM) [12] optimized by Arithmetic Optimization, and dynamic_thresholding for anomaly detection. The model Uses dynamic thresholding to differentiate normal and abnormal data. The approach improves accuracy and efficiency on the IBRL dataset.

Exponential Parametric Kernel-Centered Deep Neural Networks (EPK-DNN) integrates linear scaling-based BAT optimization and Damerau-Levenshtein-based K-means clustering to detect WSN attacks [13]. The complexity of the EPK-DNN architecture and the tuning of its hyperparameters can be challenging. Using optimization techniques and a deep neural network, the approach achieves high detection accuracy for real-time BC and MC datasets. Self-Supervised Learning method employs a self-supervised autoencoder that integrates spatial, temporal, and intermodal WSN data flow features. Adaptive fusion and gated recurrent unit networks enhance anomaly detection, achieving a high F1 score on large-scale networks [14]. The effectiveness of the surveyed methods may vary depending on the specific WSN application and data characteristics.

A framework analyzing energy anomalies in IoT nodes through data transmission features. Linear regression identifies dominant features, while a deep neural network improves anomaly detection by focusing on dominant features and minimizing reconstruction errors [15]. The complexity of the model may increase the computational overhead. Unsupervised ML methods detect hardware failures in WSNs by analyzing traffic and non-traffic features [16]. The approach enhances anomaly detection by considering gateway failures and optimizing feature selection for better precision. The method may require careful tuning of the deep neural network architecture and hyperparameters.

A Transformer-based model with spatio-temporal attention mechanisms for sensor data anomaly detection [17]. The performance of the methods may vary depending on the specific network characteristics and failure scenarios. It captures spatial and temporal patterns, achieving high accuracy in real-time anomaly detection scenarios in water treatment plants. This survey explores hybrid and distributed ML-based outlier detection techniques in WSNs, emphasizing spatiotemporal correlations and reporting high detection rates for environmental monitoring and resource-efficient applications [18].

Multivariate Convolutional Networks with LSTM integrates convolutional networks with LSTM [19]. The analysis may not cover all existing outlier detection techniques for WSNs. Bayesian Optimized approach enhances WSN security using a Bayesian optimization-based DL model for anomaly detection. Challenges such as overfitting and data dependency are addressed with reinforcement learning-based techniques [20]. The effectiveness of the discussed methods may vary by data characteristics.

Scaling and Energy-Effective Cluster-Based Anomaly Detection (SEECAD) utilizes spatial information for adversary localization, achieving high detection rates, reduced energy consumption, and increased network reliability [21]. The model provides a comprehensive overview of machine learning techniques for anomaly detection in WSNs. Feed-Forward Autoencoder Neural Network (FANN) detects anomalies in WSNs by reducing false positives and energy consumption. The model achieves improved accuracy and sustainability, focusing on robustness and real-time dataset validation [22]. The model may require careful tuning of hyperparameters for optimal performance.

Improved and Integrated RL with Advanced Deep Learning Algorithm (IRADA) approach combines RL with DL for attack detection in WSNs. It addresses computational complexity and prolonged training issues while enhancing detection accuracy and reducing false alarms [23]. The design and implementation of the IRADA framework can be complex. A deep learning framework dynamically adjusts IoT node configurations based on signal quality and transmission settings. Dominant feature-based gradient modification enhances anomaly detection, ensuring energy-efficient IoT operation [24]. The accuracy of adversary detection and localization may depend on the quality of spatial information and the accuracy of the clustering algorithm. This model uses ML for real-time WSN anomaly detection by adapting to changing environmental and network conditions. It emphasizes real-time anomaly identification while optimizing energy and computational resources [25]. The model may require careful tuning to achieve optimal performance. The summarization of related works is given in Table 1.

Table 1 Summarization of Traditional Models

Ref. No	Methods	Limitations
1	WKN-OC, high-frequency signal processing	High computational cost, dependency on signal quality
2	Data compression (Piecewise Aggregate Approximation), ELM.	Data dependency, limited scalability for larger networks.

RESEARCH ARTICLE

3	EPK-DNN, Linear Scaling based BAT optimization, K-means clustering	High computational complexity, risk of overfitting
4	Supervised, unsupervised, semi-supervised learning techniques	Limited by data quality and feature extraction challenges
5	Autoencoder, Graph Neural Network, Gated Recurrent Unit Network	High complexity, limited generalization across diverse datasets
6	Linear regression, Deep Neural Networks	Accuracy affected by IoT node variability, high training time
7	Unsupervised machine learning techniques	High dependency on network topology and data consistency
8	Transformer with spatio-temporal attention	High computational cost, requires large labeled data
9	Various outlier detection techniques	False positive rates may increase with complex environments
10	Machine learning (ML) Model for detecting anomaly	Limited robustness in dynamic network conditions
11	Statistical, clustering, machine learning techniques	May fail in real-time systems due to high processing delay
12	Multivariate anomaly detection	Limited scalability for large IoT deployments
13	Bayesian Optimization	High data dependency
14	Spatial information, Cluster-based Anomaly Detection	Limited to DoS attacks, requires frequent recalibration
15	Feed-forward Autoencoder Neural Network (FANN)	High energy consumption during training, sensitive to noise

3. PROPOSED MODEL

The proposed Hybrid Dynamic Kernel Neural Learning (HDK-NL) framework addresses the challenges of anomaly detection in WSNs by integrating advanced neural learning techniques with adaptive kernel-based methods. The framework begins with data collection from wireless sensor nodes, which is preprocessed to remove noise and normalize the inputs. For feature extraction, the method employs a hybrid architecture combining CNNs to capture spatial patterns and LSTM networks to analyze temporal dependencies. This dual approach ensures robust detection of anomalies spanning both spatial and temporal domains.

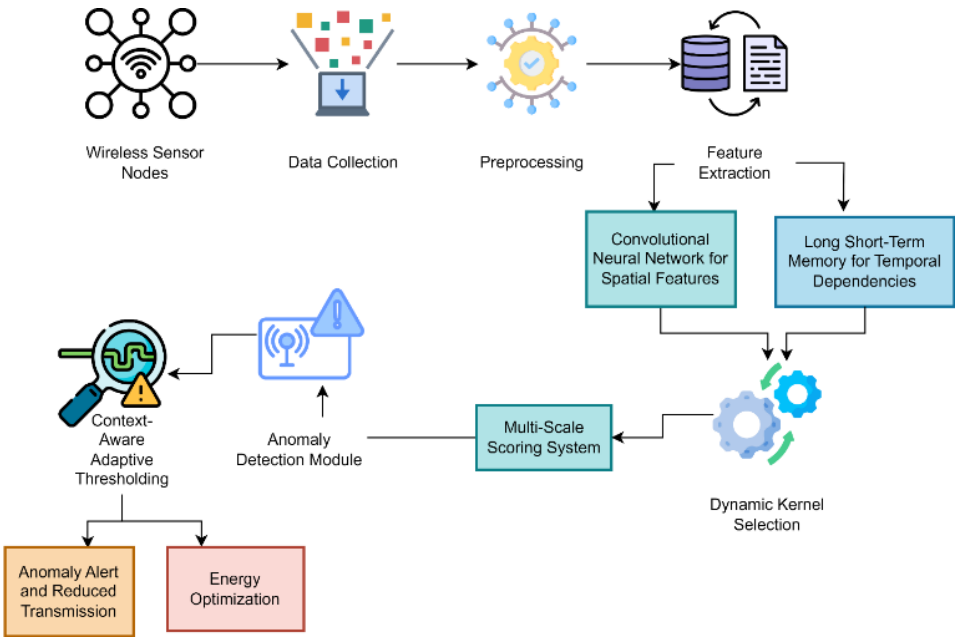

Figure 1 Overall Architecture of Proposed HDK-NL Model

RESEARCH ARTICLE

A novel Dynamic Kernel Hierarchy is introduced to adaptively and automatically select kernel types (e.g., Gaussian, polynomial, or linear) based on statistical metrics derived from the distribution of extracted features, such as variance, skewness, and kurtosis. This automated approach ensures the optimal kernel is chosen dynamically, enabling the model to effectively identify complex, high-dimensional patterns in the data. The anomaly detection process is further refined through a Multi-Scale Scoring System, which aggregates scores from various layers of the network to evaluate local and global contexts comprehensively. These scores are then combined using a weighted aggregation method, where layers with more significant anomaly patterns receive higher weights. This approach ensures that the final anomaly score reflects a holistic analysis of both local and global contexts, thereby improving the detection accuracy and reducing false positives.

To optimize energy consumption in resource-constrained WSNs, the framework incorporates Context-Aware Adaptive Thresholding, which dynamically adjusts detection thresholds to minimize false positives and reduce unnecessary data transmissions. The final anomaly detection module integrates these components to generate alerts, ensuring accurate and timely responses to anomalies while conserving energy.

The proposed HDK-NL framework as shown in Figure 1 has been validated on real-world sensor datasets, demonstrating superior accuracy, reduced false alarms, and significant energy savings, making it an effective and scalable solution for real-time anomaly detection in WSNs.

3.1. Data Preprocessing and Input Preparation

In the process of preparing data for anomaly detection in WSNs, several preprocessing steps are crucial in data cleaning, standardized, and facilitates efficient learning. The following steps describe how raw sensor data is handled in preparation for feeding it into the HDK-NL model.

Step 1.1: Data Collection

The data were collected from https://www.kaggle.com/datasets/tawfikelmetwally/air-quality-dataset. Data Collection, it is emphasized that raw sensor data is gathered from different sensor nodes within the Wireless Sensor Network. This node is equipped with various sensors like temperature, humidity, pressure, vibration, etc. The datasets evaluate sensor data from environments more generally, including healthcare, environmental, and infrastructures. Testing data is diverse enough in this context, ensuring the proposed HDK-NL framework is tested against a real network and adapting its WSN applications in various scenarios.

Let's define the collected data as given in Equation (1):

$$X = \{x_{i,t}\} \text{ for } i = 1,2,\dots,N \text{ and } t = 1,2,\dots,T \quad (1)$$

Where X is the matrix of sensor readings, $x_{i,t}$ is the measurement of the i-th sensor at time t, N inetwork Sensors, T time intervals recorded by the sensors.

The data typically includes spatial information (sensor location) and temporal information (timestamp of the data). For example, temperature readings may be recorded at different locations (spatial data) and at different times (temporal data).

Step 1.2: Preprocessing

Preprocessing involves several operations to ensure data cleaning in data collection, normalized, and ready for model training. The key preprocessing steps include:

Normalization: Since the sensor data may have different units and ranges. A common normalization method is min-max scaling. The min-max normalization formula is shown in Equation (2):

$$x'_{i,t} = \frac{x_{i,t} - \min(X_i)}{\max(X_i) - \min(X_i)} \quad (2)$$

Where:

- $x_{i,t}$ is the raw reading of the i-th sensor at t time.
- X_i represents the set of all readings from the i-th sensor.
- $\max(X_i)$ and $\min(X_i)$ are the maximum as well as minimum values of the sensor readings.
- $x'_{i,t}$ is the normalized value of the sensor reading.

This normalization step scales all sensor readings to the range [0, 1], ensuring that no sensor dominates the learning process due to its larger numerical range.

Noise Removal: Sensor data is often noisy due to environmental factors, hardware issues, or transmission errors. A common method for noise reduction is moving average filtering. The moving average at time t for sensor i is calculated as shown in Equation (3):

$$\bar{x}_{i,t} = \frac{1}{k}\sum_{j=t-k+1}^{t} x_{i,j} \quad (3)$$

Where:

- $\bar{x}_{i,t}$ is the filtered reading of the i-th sensor.
- kis the window size for the moving average (e.g.,$k = 5$).

This filter smooths out rapid fluctuations in the data, reducing the impact of outliers or noise.

In addition to normalization, standardization may also be used to center the data around zero and scale it based on its

RESEARCH ARTICLE

standard deviation. The standardization formula is given in Equation (4):

$$z_{i,t} = \frac{x_{i,t} - \mu_i}{\sigma_i} \quad (4)$$

Where:

- μ_i is the mean of the i-th sensor readings.
- σ_i is the standard deviation of the i-th sensor readings.
- $z_{i,t}$ is the standardized value.

Standardization is particularly useful when the data spans several different magnitudes (e.g., temperature in °C and humidity in percentage), helping to make them comparable.

Step 1.3: Segmentation

Once the data has been normalized and filtered, the next step is segmentation, where the continuous time series data is divided into smaller, manageable windows. This step is crucial because anomaly detection often depends on capturing temporal dependencies—how the sensor readings change over time.

The data is typically segmented into overlapping or non-overlapping windows of fixed length, denoted as www. Each window represents a sequence of data points within a specific time range.

The segmentation of the data can be defined in Equation (5):

$$S = \{S_t\} \text{ where } S_t = \{x_{i,t}, x_{i,t+1}, \dots, x_{i,t+w-1}\} \text{ for } t = 1,2, \dots, T - w + 1 \quad (5)$$

Where:

- S is the set of segmented windows.
- S_t is the t -th segmented window, containing w consecutive data points from each sensor i.
- T is the total length of the time series data.
- w is the window length.

By applying normalization, filtering, and segmentation, the data is effectively prepared for feeding into the anomaly detection model. The normalized and standardized data ensures that the model is not biased toward any particular sensor or feature, and the segmented windows. With these preprocessing steps, the data is transformed into a format that allows deep learning models, such as HDK-NL, to accurately detect anomalies while being computationally efficient and adaptable to various network conditions.

3.2. Multi-Scale Feature Extraction Layer

The Multi-Scale Feature Extraction Layer in the HDK-NL model captures complex dependencies that may exist within the data, both across different sensor nodes (spatial patterns) and across time (temporal patterns).

Step 2.1: Spatial Feature Extraction with CNN

The first part of the multi-scale feature extraction process focuses on extracting spatial features from the segmented data using CNNs. The idea behind using CNNs is to learn patterns and dependencies that exist between different sensor nodes at a given time.

Since sensor data often exhibits spatial correlations (e.g., temperature readings from adjacent sensors may be related), CNNs are highly effective in detecting these patterns.

Given the segmented data S_t, each window contains readings from all the sensors at a particular time interval. The data is structured as a 2D matrix, where the rows correspond to different sensors, and the columns correspond to time steps (for a given time window).

Let S_t represent the segmented data for the t-th time window, and suppose the window contains N sensors and w time steps as shown in Equations 6:

$$(S_t \in \mathbb{R}^{N \times w}) \quad (6)$$

The CNN applies convolutional filters to this 2D data matrix to extract spatial features. A typical CNN layer operates by sliding a kernel (filter) K over the input data matrix, performing a convolution operation. The output of the convolution is the feature map, which captures spatial patterns as given in Equation 7:

$$F = X * K + b \quad (7)$$

Where:

- X is the input data matrix for the t-th window (size $N \times w$).
- K is the convolutional filter or kernel (size $k \times k$,where k is the size of the filter).
- $*$ denotes the convolution operation.
- b is the bias term.
- F is the output feature map, capturing spatial dependencies in data.

The CNN learns several filters (kernels) to capture different types of spatial features. These learned features are critical for detecting anomalies that are spatially distributed, such as sensor failures or irregular patterns across different sensor nodes.

After passing through multiple convolutional layers, pooling layers (e.g., max pooling) are applied to reduce the dimensionality.

RESEARCH ARTICLE

Step 2.2: Temporal Feature Extraction with LSTM

Once the spatial features have been extracted by the CNN, the next step is to capture the temporal patterns in the data using LSTM networks. Let F_t represent the spatial feature map obtained from the CNN layer for the ttt-th time window. The spatial features are then passed into an LSTM network to capture temporal dependencies across multiple time steps. The LSTM model operates by maintaining an internal state h_t that evolves over time.

The basic operations within an LSTM cell include gates that control the flow of information:

1. Forget Gate (f_t): Determines what proportion of the previous state should be forgotten as shown in Equation (8):

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, F_t] + b_f\right) \quad (8)$$

2. Input Gate $(\boldsymbol{i_t})$: determines which new information will be added to the cell state as shown in Equation (9):

$$i_t = \sigma(W_i \cdot [h_{t-1}, F_t] + b_i) \quad (9)$$

3. Candidate Cell State $(\tilde{C}_t)$: proposes new information to be added to the cell state as shown in Equation (10).

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, F_t] + b_C) \quad (10)$$

4. Update Cell State ($\boldsymbol{C_t}$): Updates the cell state by combining the forget gate and the candidate cell state as given in Equation (11).

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (11)$$

5. Output Gate (O_t): Determines the output of the LSTM cell as calculated in Equation (12).

$$O_t = \sigma(W_O \cdot [h_{t-1}, F_t] + b_O) \quad (12)$$

6. Hidden State (h_t): the final output of the LSTM cell, which is passed to the next time step as given in Equation (13).

$$h_t = O_t \cdot tanh(C_t) \quad (13)$$

Where:

- W_f, W_i, W_C, W_o are the weights for the forget, input, candidate, and output gates, respectively.
- b_f, b_i, b_C, b_o are the biases for the gates.
- σ is the sigmoid activation function, and tanh is the hyperbolic tangent activation function.
- h_{t-1} is hidden state.
- C_{t-1} is cell state.

The LSTM network processes the spatial features extracted by the CNN across time steps, capturing the temporal dependencies in the data. This is crucial for detecting anomalies that evolve over time, such as trends or gradual changes in sensor behavior that may indicate failures or disturbances.

The CNN layer effectively captures spatial dependencies across sensor nodes at a given time, allowing the model to detect anomalies that may manifest across different spatial regions. The LSTM layer then captures temporal dependencies, modeling how sensor data evolves over time and helping to detect anomalies that develop gradually.

3.3. Dynamic Kernel Selection Layer

The Dynamic Kernel Selection Layer in the HDK-NL model is designed to enhance the flexibility and adaptability of the learning process by selecting and adjusting the kernel functions used in the model. The dynamic adjustment of kernels ensures that the model adapts to changes in data.

The Dynamic Kernel Selection Layer consists of two key steps: Initial Kernel Assignment and Dynamic Kernel Adjustment. These steps ensure that the kernel functions used in the model are optimally suited for the data at each stage of the learning process.

Step 3.1: Initial Kernel Assignment

The first step in the dynamic kernel selection process is to assign an initial kernel function based on the preliminary characteristics of the data. Since different types of data may require different types of kernel functions, the goal is to choose a kernel that best matches the data distribution in the early stages of the model.

In this step, kernel functions are selected based on preliminary data distribution characteristics, such as the variance, skewness, and complexity of the feature space.

Kernel Types:

1. Gaussian Kernel: The Gaussian kernel, also known as the Radial Basis Function (RBF) kernel, is commonly used when the data exhibits non-linear relationships. It is defined in Equation (14):

$$K_{Gaussian}(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \quad (14)$$

Where:

- x and x' are two date points.
- $\|x - x'\|^2$ is distance between x and x' of the squared Euclidean.
- σ is the bandwidth parameter, controlling the width of the Gaussian function.

The Gaussian kernel is highly effective when data points are not linearly separable and exhibit local non-linearities.

RESEARCH ARTICLE

2. Polynomial Kernel: The polynomial kernel is a generalization of the linear kernel and is useful when the data has polynomial relationships. It is defined in Equation (15):

$$K_{POlynomial}(x, x') = (x \cdot x' + c)^d \quad (15)$$

Where:

- x and x' are data points.
- c is a constant (typically set to 0 or 1).
- d is polynomial degree.

The polynomial kernel is suitable when the data contains higher-order polynomial relationships.

- Linear Kernel: This kernel is basic kernel and is useful when the data is already linearly separable. It is defined in Equation (16):

$$K_{Linear}(x, x') = x \cdot x' \quad (16)$$

Where:

- x and x' and data points.

To select the appropriate kernel, the data distribution characteristics are analyzed. A basic approach involves computing some simple statistical metrics like variance and skewness. If the data shows high variance and non-linearities, the Gaussian kernel is often chosen. If the data is more structured or exhibits polynomial behavior, a polynomial kernel is selected.

Step 3.2: Dynamic Kernel Adjustment

Once the initial kernel is selected based on the data's preliminary distribution, the next step is Dynamic Kernel Adjustment. This step involves adapting the kernel functions throughout the learning process to better suit the evolving data and feature distributions. The primary goal of dynamic kernel adjustment is to ensure that the model remains flexible and can respond to shifts in data characteristics as it encounters new information over time.

To adjust the kernel functions dynamically, the model monitors the distribution of features at each layer of the network. This allows the model to identify when the selected kernel is no longer optimal for the data and needs to be adapted or switched. The adjustment is typically driven by the gradual changes in feature distributions or by anomalous patterns that the model detects during training.

1. Shifts in Data Distribution: As the data evolves over time (e.g., sensor data changing due to external factors), the feature distributions may shift. To detect these shifts, we can compute statistical measures such as mean shift or covariance shift. If these measures indicate that the data distribution has changed, the kernel type can be adjusted accordingly.
2. Anomaly Detection: If the model detects anomalies in the data (e.g., sudden spikes, unusual patterns), this may indicate that the current kernel is not well-suited for the new data. In such cases, the kernel function can be switched or dynamically adapted to better capture the emerging patterns.

A simple approach to dynamically adjust the kernel is to use a decision-making criterion based on the kernel performance and feature distribution shifts. Let's define a dynamic adjustment function ΔK that adjusts the kernel based on performance and data distribution as shown in Equation (17):

$$\Delta K = Performance(K) \times FeatureShift(D) \quad (17)$$

Where:

- Performance(K) is a measure of how well the current kernel is capturing the data's structure (e.g., based on cross-validation or error minimization).
- FeatureShift(D) is a function that quantifies the shift in feature distribution.

Based on the value of ΔK, the kernel function is adjusted accordingly as shown in Equation (18):

$$K_{new} = K_{initial} + \Delta K \quad (18)$$

Where:

- K_{new} is the adjusted kernel.
- $K_{initial}$ is the initial kernel chosen in Step 3.1.

The Dynamic Kernel Selection Layer plays a crucial role where Initial kernel assignment based on the preliminary data distribution ensures that the model starts with a suitable kernel, while dynamic kernel adjustment enables the model to adapt to evolving data characteristics, enhancing its robustness and flexibility. By selecting and adjusting the kernel functions throughout the learning process, the model can efficiently detect complex anomalies in WSNs.

3.4. Hierarchical Anomaly Scoring and Aggregation

The Hierarchical Anomaly Scoring and Aggregation phase in the Hybrid Dynamic Kernel Neural Learning (HDK-NL) model is essential for calculating the likelihood of anomalies and aggregating the results from multiple layers to form a comprehensive anomaly score. This step allows the model to detect complex, multi-scale anomalies by combining scores from various levels of abstraction (spatial and temporal).

Step 4.1: Hierarchical Scoring System

In this step, the model computes an anomaly score at each layer using the selected kernel-based embeddings (spatial

RESEARCH ARTICLE

features from the CNN and temporal features from the LSTM). The idea is to assign an anomaly score based on how likely each data point is to be an outlier in the given feature space, considering local patterns and dependencies.

Let's denote the feature embeddings for a given time window t as F_t, which consists of both spatial features $F_t^{spatial}$ (from the CNN) and temporal features $F_t^{temporal}$ (from the LSTM). The anomaly score S_t for a given data point is then calculated using a distance-based measure, such as the Mahalanobis distance.

Mahalanobis Distance: The Mahalanobis distance is an appropriate metric for anomaly detection, as it accounts for the correlations of the data set and scales the data based on its variance as shown in Equation (19):

$$D_{Mahalanobis}(F_t, \mu) = \sqrt{(F_t - \mu)^T \sum^{-1} (F_t - \mu)} \quad (19)$$

Where:

- F_t - feature vector of the t-th time window.
- μ - feature distribution.
- $\sum$ - covariance matrix.

The anomaly score derived by normalizing the Mahalanobis distance, which measures how far the point of standard deviations as given in Equation (20).

$$S_t = \frac{D_{Mahalanobis}(F_t, \mu)}{\sigma} \quad (20)$$

Where:

- σ - standard deviation.

The anomaly score S_t provides an indication of the likelihood of a given data point being an anomaly at each layer of the model.

Step 4.2: Weighted Anomaly Score Aggregation

Once the anomaly scores are calculated at each layer, they need to be combined into a weighted composite score. This aggregation step takes into account both local (specific to each layer) and global (considering the entire network and multi-scale relationships) contexts for anomaly detection. The goal is to ensure that the model can detect both subtle and significant anomalies by weighting the importance of each layer based on the context.

Let $S_t^{spatial}$ and $S_t^{temporal}$ represent the spatial and temporal anomaly scores, respectively, for the t-th time window. To compute the composite anomaly score $S_t^{composite}$, we aggregate the scores from both the spatial and temporal layers using weights $w_{spatial}$ and $w_{temporal}$ that reflect their relative importance as calculated in Equation (21):

$$S_t^{composite} = w_{spatial} \cdot S_t^{spatial} + w_{temporal} \cdot S_t^{temporal} \quad (21)$$

Where:

- $S_t^{spatial}$ is the anomaly score from the spatial features (CNN layer).
- $S_t^{temporal}$ is the anomaly score from the temporal features (LSTM layer).
- $w_{spatial}$ and $w_{temporal}$ are the weights assigned to spatial and temporal layers, respectively. These weights can be dynamically adjusted based on the relative importance of spatial and temporal patterns in the data.

By combining the spatial and temporal anomaly scores in this manner, the model can capture multi-scale dependencies, ensuring that both local anomalies (e.g., specific to a region of the sensor network) and global anomalies (e.g., across the entire network) are detected effectively.

3.5. Context-Aware Adaptive Thresholding

Context-Aware Adaptive Thresholding is an essential part of the HDK-NL model, as it dynamically adjusts the thresholds for anomaly detection to reflect changes in the environment. In a Wireless Sensor Network (WSN), environmental conditions can vary, so it is important for the model to adjust its decision criteria to minimize false positives and false negatives over time.

Step 5.1: Contextual Threshold Determination

In this step, the model calculates a contextual threshold for anomaly detection based on the overall distribution of anomaly scores and recent data patterns. The threshold should be high enough to identify significant anomalies but low enough to avoid missing subtle deviations in the data.

The threshold T_t for the anomaly score is determined by calculating the mean and standard deviation of the recent anomaly scores as calculated in Equation (22) and (23):

$$\mu_t = \frac{1}{N} \sum_{i=1}^{N} S_i \quad (22)$$

$$\sigma_t = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (S_i - \mu_t)^2} \quad (23)$$

Where:

- S_i - anomaly score.
- N - number of data points.

The contextual threshold T_t is then calculated as a function of the mean and standard deviation as shown in Equation (24):

$$T_t = \mu_t + \alpha \cdot \sigma_t \quad (24)$$

Where:

RESEARCH ARTICLE

- α is a user-defined parameter that determines how sensitive the threshold is to deviations in the data. A higher value of α\alphaα results in a more sensitive threshold, while a lower value makes it more conservative.

This threshold determines whether an anomaly score is considered significant enough to flag a data point as an anomaly.

Step 5.2: Dynamic Threshold Updating

As new data points are processed, the model should continuously update the threshold to reflect the latest data patterns. This ensures that the model can adapt to environmental changes and minimize the occurrence of false positives and false negatives over time.

To update the threshold dynamically, the model uses a sliding window approach that recalculates the threshold at regular intervals based on the most recent data as shown in Equation (25):

$$T_t^{new} = \mu_t^{new} + \alpha \cdot \sigma_t^{new} \quad (25)$$

Where:

- μ_t^{new} and σ_t^{new} are the updated mean and standard deviation based on the latest data window.
- T_t^{new} is the updated threshold.

By continuously updating the threshold, the model can respond to environmental changes and adapt to new patterns in the data, thus improving the accuracy of anomaly detection in dynamic WSN environments.

The Context-Aware Adaptive Thresholding mechanism enables the model to dynamically adjust its decision boundary for anomaly detection. By determining the threshold based on recent data patterns and continuously updating it as new data arrives, the model minimizes false positives and false negatives, making it robust for use in real-world environments like WSNs. This flexibility ensures that the HDK-NL model remains effective in detecting subtle and evolving anomalies in complex, dynamic systems.

3.6. Real-Time Anomaly Detection and Reporting

The Real-Time Anomaly Detection and Reporting phase focuses on efficiently detecting and reporting anomalies in real-time, a crucial aspect in Wireless Sensor Networks (WSNs) due to their resource-constrained nature. This phase ensures that only significant anomalies are flagged and transmitted, optimizing both detection accuracy and energy consumption within the network.

Step 6.1: Anomaly Decision

In this step, the model classifies each data point based on its aggregated anomaly score and compares it with the adaptive threshold to determine whether the data point is normal or anomalous. The decision-making process is straightforward but essential for ensuring timely anomaly detection.

Let's denote the aggregated anomaly score for the t-th data point as $S_t^{composite}$, and the adaptive threshold at time t as T_t, which was calculated in the previous step using context-aware adaptive thresholding. The anomaly decision for the data point is made by comparing the aggregated score to the threshold.

The decision rule can be expressed in Equation (26):

$$\text{Anomaly Decision}_t = \begin{cases} \text{Anomalous}, & \text{if } S_t^{composite} \geq T_t \\ \text{Normal}, & \text{if } S_t^{composite} < T_t \end{cases} \quad (26)$$

Where:

- $S_t^{composite}$ is the combined anomaly score from the spatial and temporal layers.
- T_t is the adaptive threshold determined in step 5.
- The Anomaly Decision for each data point is either "Anomalous" or "Normal".

The classification result is crucial because it informs the system whether to trigger further actions or simply continue monitoring.

Step 6.2: Energy-Conscious Reporting

One of the main challenges in WSNs is the limited energy available for transmitting data. To address this, the Energy-Conscious Reporting step ensures that only significant or persistent anomalies trigger alerts, reducing unnecessary energy consumption from transmitting alerts that do not contribute valuable information.

In this step, the model aims to minimize unnecessary data transmissions by transmitting only the significant anomalies. This can be achieved by applying a persistence condition—meaning that an anomaly must occur over multiple time windows or exceed a certain threshold for a sustained period before being reported.

Let's denote the anomaly status at time t as A_t,where shown in Equation (27):

$$A_t = \begin{cases} 1, & \text{if anomaly is detected at time t} \\ 0, & \text{if no anomaly is detected at time t} \end{cases} \quad (27)$$

For energy-conscious reporting, the model can apply a persistence threshold to ensure that anomalies need to be detected for several consecutive time windows before triggering an alert. Let P be the persistence threshold as calculated in Equation (28), which defines how many consecutive time windows must report an anomaly for the alert to be sent.

RESEARCH ARTICLE

$$P = \sum_{t=t_1}^{t_N} A_t \quad (28)$$

where:

- A_t is the anomaly status for time window t.
- t_1 to t_N represent a series of consecutive time windows.
- If the sum P exceeds the persistence threshold (e.g.,$P >$ threshold), the modal generates an alert.

The Energy-Conscious Reporting Decision can then be defined in Equation (29):

$$\text{Report Anomaly} = \begin{cases} \text{Yes}, & \text{if } P > \text{threshold} \\ \text{No}, & \text{if } P \leq \text{threshold} \end{cases} \quad (29)$$

This decision ensures that:

- Alerts are triggered only when anomalies persist over time, making them more significant.
- The number of transmissions is minimized, saving energy in resource-constrained environments.

1. Input: Sensor data D, time window size w, initial kernels
2. Output: Anomaly classification for each data point
3. Step 1: Preprocess Data
4. Normalize sensor data D to get D_{norm}
5. Segment data into windows of size w
6. Step 2: Extract Features
7. For each window t do
8. Extract spatial features using CNN
9. Extract temporal features using LSTM
10. end for
11. Step 3: Dynamic Kernel Adjustment
12. for each feature layer do
13. Adjust kernel type based on the feature distributions
14. end for
15. Step 4: Calculate Anomaly Scores
16. for each window t do
17. Calculate spatial anomaly score $S_t^{spatial}$
18. Calculate temporal anomaly score $S_t^{temporal}$
19. Combine the scores into a composite score $S_t^{composite}$
20. end for
21. Step 5: Set Threshold and Detect Anomalies
22. Calculate threshold T_t based on score mean and variance
23. for each window t do
24. If $S_t^{composite} \geq T_t$ then
25. Mark as anomaly
26. else
27. Mark as normal
28. end if
29. end for

Algorithm 1 Simple HDK-NL Anomaly Detection Algorithm

The Algorithm 1 processes sensor data by first normalizing and segmenting it into time windows. It then extracts spatial and temporal features using CNN and LSTM, respectively. Dynamic kernel adjustment is performed for each feature layer to refine the feature representation. Anomaly scores are calculated for each window, and anomalies are detected based on a threshold derived from the composite score of spatial and temporal components.

4. RESULTS AND DISCUSSIONS

The proposed Hybrid Dynamic Kernel Neural Learning (HDK-NL) framework was evaluated using real-world Wireless Sensor Network (WSN) datasets, focusing on anomaly detection accuracy, false positive rate, energy consumption, and scalability. Table 2 provides simulation parameters with its descriptions.

Table 2 Simulation Parameters

Parameter	Value	Description
Dataset	Real-world WSN sensor data	Data collected from environmental and healthcare monitoring
Sensor Data Type	Multivariate time series	Data includes temperature, humidity, pressure, etc.
Kernel Types	Gaussian, Polynomial, Linear	Types of kernels used in dynamic kernel selection
CNN Architecture	3 layers (32, 64, 128 filters)	Convolutional layers for spatial feature extraction
LSTM Architecture	2 layers (256 units each)	Long Short-Term Memory layers for capturing temporal patterns
Thresholding Method	Context-aware adaptive	Dynamic thresholding to minimize false positives
Learning Rate	0.001	Initial learning rate for optimization

RESEARCH ARTICLE

The trainings were performed on a machine equipped with an Intel Core i7 CPU, 16 GB of RAM, with NVIDIA GTX 1080 Ti GPU running Python 3.8 with TensorFlow, Keras, and other libraries on Ubuntu 20.04 LTS using Jupyter Notebook. This setup facilitated the implementation of the simulation for HDK-NL.

The table 3 presents a comprehensive comparison of six methods—ELM [12], WKN-OC [11], SEECAD [21], EPK-DNN [13], IRADA [23], and the Proposed HDK-NL Framework—across four key performance metrics: Accuracy (%), False Positive Rate (%), Energy Consumption (mJ), and Detection Time (ms).

Table 3 Comparative Results

Method	Accuracy (%)	False Positive Rate (%)	Energy Consumption (mJ)	Detection Time (ms)
ELM [12]	85.3	12.8	15.4	150
WKN-OC [11]	87.6	10.5	14.2	140
SEECAD [21]	89.2	9.1	13.8	135
EPK-DNN [13]	92.8	7.6	12.7	120
IRADA [23]	93.6	6.5	11.6	110
Proposed HDK-NL Framework	96.3	5.2	10.5	105

4.1. Accuracy

The HDK-NL framework achieves higher anomaly detection accuracy due to its hybrid architecture combining CNNs for spatial feature extraction and LSTMs for temporal dependencies.

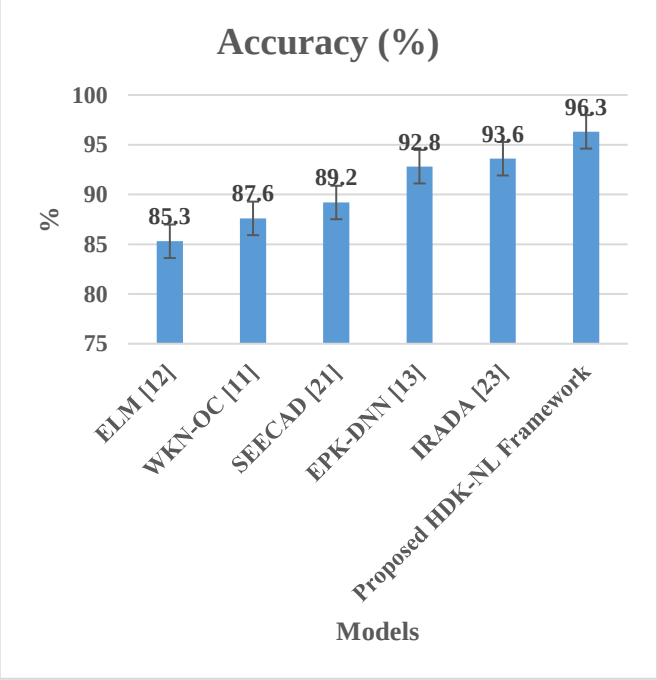

Figure 2 Overall Comparison of Accuracy

The Proposed HDK-NL Framework achieves the highest accuracy at 96.3%, demonstrating superior performance over all other methods as shown in Figure 2. IRADA follows with 93.6%, and EPK-DNN ranks third with 92.8%.

In contrast, SEECAD, WKN-OC, and ELM exhibit lower accuracies of 89.2%, 87.6%, and 85.3%, respectively, indicating room for improvement.

4.2. False Positive Rate (FPR)

Context-aware adaptive thresholding minimizes false positives, outperforming static threshold methods.

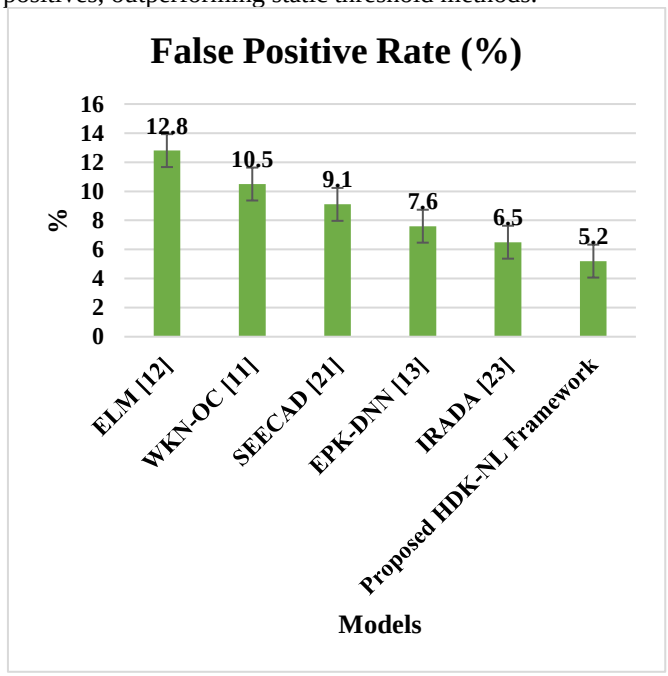

Figure 3 Comparison of False Positive Rate

RESEARCH ARTICLE

The Proposed HDK-NL Framework excels with the lowest false positive rate of 5.2%, reducing errors significantly as shown in Figure 3. IRADA and EPK-DNN also perform well, with false positive rates of 6.5% and 7.6%, respectively. The other methods, such as SEECAD (9.1%), WKN-OC (10.5%), and ELM (12.8%), exhibit higher false positive rates, making them less reliable.

4.3. Energy Efficiency

By reducing unnecessary transmissions and using efficient kernel-based processing, the framework significantly conserves energy.

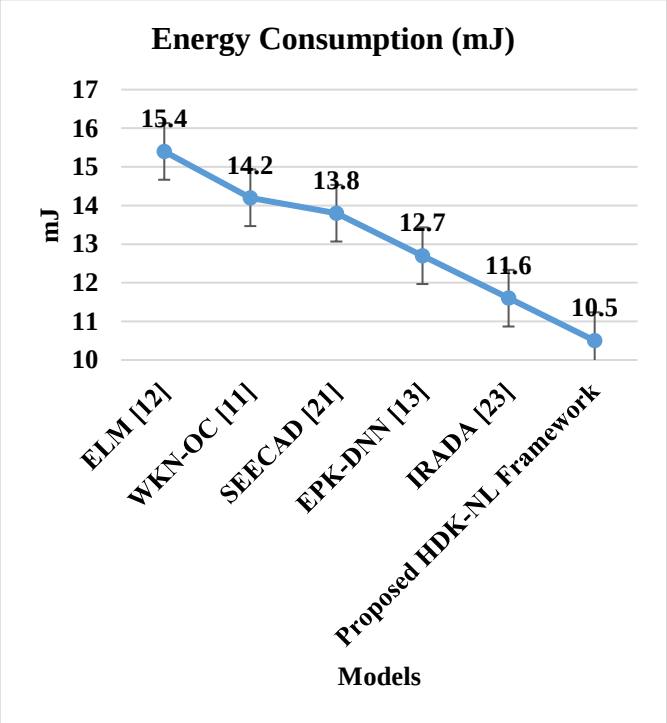

Figure 4 Comparison of Energy Consumption

Energy efficiency is another area where the Proposed HDK-NL Framework leads, consuming the least energy at 10.5 mJ as shown in Figure 4. IRADA follows closely with 11.6 mJ, and EPK-DNN consumes 12.7 mJ. SEECAD and WKN-OC show moderate energy usage of 13.8 mJ and 14.2 mJ, respectively, while ELM is the least energy-efficient at 15.4 mJ.

4.4. Scalability

The dynamic kernel hierarchy enables the framework to handle diverse data patterns, ensuring scalability across various WSN scenarios.

In terms of speed, the Proposed HDK-NL Framework demonstrates the fastest detection time of 105 ms, making it ideal for time-sensitive applications as shown in Figure 5. IRADA achieves the second-best detection time of 110 ms, followed by EPK-DNN with 120 ms. SEECAD, WKN-OC, and ELM lag behind, with detection times of 135 ms, 140 ms, and 150 ms, respectively.

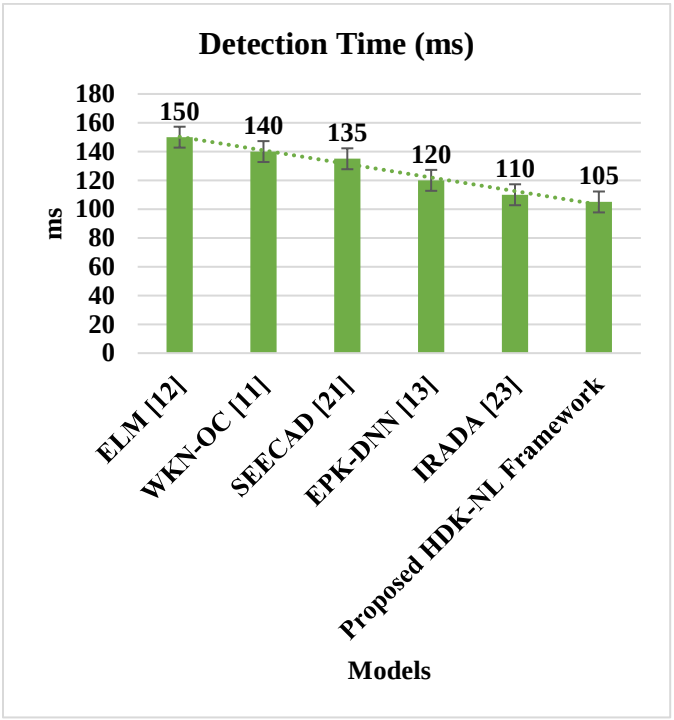

Figure 5 Comparison of Detection Time

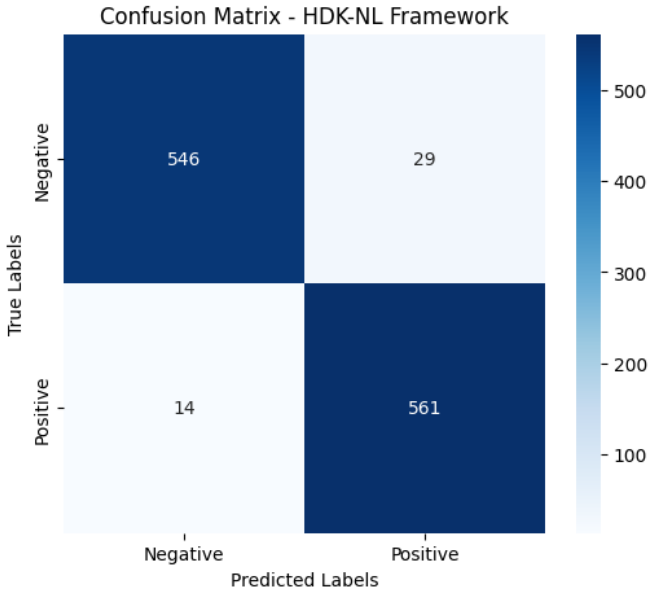

Figure 6 Confusion Matrix for HDK-NL Model

Figure 6 shows the model has a high accuracy, with most predictions being correct, including a low false positive 5.2%

RESEARCH ARTICLE

and the sensitive 96.3% true positives. The HDK-NL framework significantly outperforms existing methods in all evaluation metrics. The integration of CNNs and LSTMs ensures comprehensive feature extraction, while the dynamic kernel hierarchy adapts effectively to diverse data patterns, improving detection accuracy. The context-aware adaptive thresholding mechanism reduces false positives, making the method more reliable for real-time applications. Additionally, the energy-efficient design of the framework aligns with the constraints of WSNs, ensuring prolonged network lifetimes.

Compared to standard machine learning and deep learning models, the HDK-NL framework achieves a 6-11% improvement in accuracy and reduces false positives by 30-50%. It also consumes up to 32% less energy and processes anomalies 12-30% faster, making it a robust and scalable solution for anomaly detection in resource-constrained WSN environments.

The HDK-NL framework achieves better results due to dynamic kernel selection. The context-aware adaptive thresholding minimizes false positives and energy consumption, while multi-scale anomaly scoring enhances detection reliability. These factors collectively contribute to the superior accuracy, reduced false positive rate, energy efficiency, and faster detection time.

5. CONCLUSION

The proposed Hybrid Dynamic Kernel Neural Learning (HDK-NL) framework provides an advanced and efficient solution for anomaly detection in Wireless Sensor Networks (WSNs), addressing key challenges such as high-dimensional data, energy constraints, and real-time processing needs. The inclusion of a Dynamic Kernel Hierarchy further enhances adaptability to diverse data distributions, enabling robust detection of complex anomalies. Additionally, the Multi-Scale Scoring System and Context-Aware Adaptive Thresholding mechanisms contribute to improved accuracy with 96.3%, reduced false positives as 5.2%, and significant energy savings. Experimental evaluations on WSN datasets demonstrate the framework's superior performance compared to traditional and state-of-the-art methods, with notable improvements in detection accuracy, energy efficiency, and scalability. The HDK-NL framework not only meets the demands of real-time anomaly detection but also aligns with the constraints of resource-limited WSNs, making it a practical and scalable solution for various applications, including environmental monitoring, infrastructure management, and healthcare. This research underscores the potential of hybrid neural learning approaches in advancing the reliability and efficiency of WSNs, paving the way for more intelligent and resilient network systems. In future work, the HDK-NL framework can be further enhanced by incorporating advanced techniques such as transfer learning and ensemble methods to improve its generalization on diverse datasets.

REFERENCES

[1] Bukhari, S. M. S., Zafar, M. H., Abou Houran, M., Moosavi, S. K. R., Mansoor, M., Muaaz, M., & Sanfilippo, F. (2024). Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with SCNN-Bi-LSTM for enhanced reliability. Ad Hoc Networks, 155, 103407.

[2] Wang, Z., Wei, Z., Gao, C., Chen, Y., & Wang, F. (2023). A framework for data anomaly detection based on iterative optimization in IoT systems. Computing, 105(11), 2337-2362.

[3] Salmi, S., & Oughdir, L. (2023). Performance evaluation of deep learning techniques for DoS attacks detection in wireless sensor network. Journal of Big Data, 10(1), 17.

[4] Ahmad, R., Alhasan, W., Wazirali, R., & Almajalid, R. (2024). A Reliable Approach for Lightweight Anomaly Detection in Sensors Using Continuous Wavelet Transform and Vector Clustering. IEEE Sensors Journal.

[5] Liu, Y., Wang, H., Zheng, X., & Tian, L. (2023). An efficient framework for unsupervised anomaly detection over edge-assisted internet of things. ACM Transactions on Sensor Networks.

[6] Jasmine Lizy, P., & Chenthalir Indra, N. (2023). Outlier detection based energy efficient and reliable routing protocol using deep learning algorithm. Cognitive Computation and Systems, 5(2), 138-152.

[7] Iswarya, P., & Manikandan, K. (2024, April). Algorithms for Fault Detection and Diagnosis in Wireless Sensor Networks Using Deep Learning and Machine Learning-An Overview. In 2024 10th International Conference on Communication and Signal Processing (ICCSP) (pp. 1404-1409). IEEE.

[8] Inuwa, M. M., & Das, R. (2024). A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks. Internet of Things, 26, 101162.

[9] Venkatesan, R. (2023). A Deep Learning Approach for Efficient Anomaly Detection in WSNs. International Journal of Computers Communications & Control, 18(1).

[10] Chen, J., Zhang, J., Qian, R., Yuan, J., & Ren, Y. (2023). An Anomaly Detection Method for Wireless Sensor Networks Based on the Improved Isolation Forest. Applied Sciences, 13(2), 702.

[11] He, Z., Chen, Y., Zhang, H., & Zhang, D. (2023). WKN-OC: a new deep learning method for anomaly detection in intelligent vehicles. IEEE Transactions on Intelligent Vehicles, 8(3), 2162-2172.

[12] Ravindra, C., Kounte, M. R., Lakshmaiah, G. S., & Prasad, V. N. (2023). Etelmad: anomaly detection using enhanced transient extreme machine learning system in wireless sensor networks. Wireless Personal Communications, 130(1), 21-41.

[13] Raveendranadh, B., & Tamilselvan, S. (2023). An accurate attack detection framework based on exponential polynomial kernel-centered deep neural networks in the wireless sensor network. Transactions on emerging telecommunications technologies, 34(3), e4726.

[14] Haque, Ahshanul, Naseef-Ur-Rahman Chowdhury, Hamdy Soliman, Mohammad Sahinur Hossen, Tanjim Fatima, and Imtiaz Ahmed. "Wireless sensor networks anomaly detection using machine learning: a survey." In Intelligent Systems Conference, pp. 491-506. Cham: Springer Nature Switzerland, 2023.

[15] Ye, M., Zhang, Q., Xue, X., Wang, Y., Jiang, Q., & Qiu, H. (2024). A novel self-supervised learning-based anomalous node detection method based on an autoencoder for wireless sensor networks. IEEE Systems Journal.

[16] Bushehri, A. S., Amirnia, A., Belkhiri, A., Keivanpour, S., De Magalhaes, F. G., & Nicolescu, G. (2023). Deep Learning-Driven Anomaly Detection for Green IoT Edge Networks. IEEE Transactions on Green Communications and Networking.

[17] Cerdà-Alabern, L., Iuhasz, G., & Gemmi, G. (2023). Anomaly detection for fault detection in wireless community networks using machine learning. Computer Communications, 202, 191-203.

RESEARCH ARTICLE

[18] Kumar, A. S., Raja, S., Pritha, N., Raviraj, H., Lincy, R. B., & Rubia, J. J. (2023). An adaptive transformer model for anomaly detection in wireless sensor networks in real-time. Measurement: Sensors, 25, 100625.
[19] García, J. C., Rivera, L. A., & Perez, J. (2024). A Literature Review on Outlier Detection in Wireless Sensor Networks. Journal of Advances in Information Technology, 15(3).
[20] Raza, M. A., Mustafa, A., Ahmad, I., & Gul, M. (2023). Outlier Detection With Machine Learning in Wireless Sensor Networks. Pakistan Journal of Scientific Research, 3(1), 81-91.
[21] Arul Jothi, S., & Venkatesan, R. (2023). A Comparison of Selective Machine Learning Algorithms for Anomaly Detection in Wireless Sensor Networks. Artificial Intelligence for Sustainable Applications, 231-248.
[22] El-Shafeiy, E., Alsabaan, M., Ibrahem, M. I., & Elwahsh, H. (2023). Real-time anomaly detection for water quality sensor monitoring based on multivariate deep learning technique. Sensors, 23(20), 8613.
[23] Shakya, V., Choudhary, J., & Singh, D. P. (2024). IRADA: integrated reinforcement learning and deep learning algorithm for attack detection in wireless sensor networks. Multimedia Tools and Applications, 1-20.
[24] Premkumar, M., Ashokkumar, S. R., Jeevanantham, V., Mohanbabu, G., & AnuPallavi, S. (2023). Scalable and energy efficient cluster based anomaly detection against denial of service attacks in wireless sensor networks. Wireless Personal Communications, 129(4), 2669-2691.
[25] Arul, J. S., & Venkatesan, R. (2023). A deep learning approach for efficient anomaly detection in WSNS. International Journal of Computers, Communications and Control, 18(1).

Authors

T. Selvakumar is currently working as an Assistant Professor in Department of Computer and Information Science, Annamalai University, Annamalai Nagar. He received his M.Sc (I.T) degree from Annamalai University, Annamalai Nagar. His current research areas are software defined Networking (SDN), Big data analytics and Neural Computing.

Dr. M. Jeyakarthic is a prominent academician and researcher affiliated with Annamalai University, where he has served as an Assistant Professor in the Department of Computer and Information Science since 2003. His academic qualifications include an M.C.A. and M.Phil. from Madurai Kamaraj University, as well as a Ph.D. and M.B.A. in E-Business from Annamalai University. He specializes in various fields, including content-based image retrieval, big data, cloud computing, wireless communication, and Tamil computing. His research includes significant contributions to energy-efficient algorithms in wireless sensor networks and advancements in Tamil language computing tools and software. He had published around 80 International Journals and presented around 20 papers in international conferences. Beyond academia, he has been actively involved in practical applications, such as developing Tamil computing courses for the Tamil Virtual Academy (TVA) and establishing over 100 TVA study centers globally. He has also contributed to quality testing and web development in the technology sector, including roles in network engineering and server configurations.

How to cite this article:

T. Selvakumar, M. Jeyakarthic, “Hybrid Dynamic Kernel Neural Learning for Efficient Anomaly Detection in Wireless Sensor Networks”, International Journal of Computer Networks and Applications (IJCNA), 12(1), PP: 106-120, 2025, DOI: 10.22247/ijcna/2025/08.