Revolutionising IoT Network Security By Assessing ML Localisation Techniques Against Jamming Attacks

Iacovos Ioannou*†, Michael Savva*, Marios Raspopoulos‡,§, Christophoros Christophorou*†, Vasos Vassiliou*†
*University of Cyprus, Department of Computer Science, Nicosia, Cyprus
†CYENS Centre of Excellence, Nicosia, Cyprus
‡University of Central Lancashire Cyprus, Larnaka, Cyprus
§INSPIRE Research Centre, Cyprus

Abstract*—The Internet of Things (IoT) revolutionises data flow management by interconnecting various devices, from simple sensors to complex systems, for varied applications, including smart homes and industrial automation. However, this extensive integration also renders IoT networks vulnerable to cyber threats, mainly jamming attacks, which can disrupt wireless communications and risk total network failure, significantly impacting critical sectors such as healthcare and industrial automation. This research introduces a novel approach to IoT network security, transitioning from conventional detection methods to a refined strategy emphasising the accurate localisation of jamming sources. Integrating machine learning with network security to counter jamming attacks establishes a foundation for future exploration. It effectively utilises Fuzzy Logic-based Intrusion Detection Systems (FLIDS) for the initial detection of jamming threats, addressing a vital need for heightened security in IoT networks. Central to our research is adopting advanced machine learning models, including Recurrent Neural Network - Bidirectional Long Short-Term Memory (RNN-B-LSTM), Temporal Convolutional Network (TCN), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). These models are carefully selected for their capacity to process complex data patterns and suitability for real-time IoT environments. We conduct a thorough evaluation of these models using various metrics. Consequently, the RNN-B-LSTM model, in particular, demonstrates exceptional accuracy in jammer localisation.

***Index Terms*—ML Localisation, ML Jamming Localisation, FLIDS, RNN-B-LSTM, TCN, RNN, CNN, Jamming identification, Jamming Localisation**

I. Introduction

The Internet of Things (IoT) represents a network of interconnected devices, from simple sensors to complex systems, integrated into various aspects of daily life, such as smart homes and industrial systems. However, the widespread deployment of IoT devices and their inherent design limitations, like limited processing capacity and energy efficiency focus, make them susceptible to cyberattacks, mainly jamming attacks. These attacks, which disrupt wireless connectivity by flooding networks with radio interference, exploit the unrestricted nature of wireless channels, emphasising the need for sophisticated security solutions. Traditional methods often must catch up against such advanced threats, highlighting the urgency for effective defence mechanisms.

The impact of jamming on IoT networks is profound, leading to disruptions or complete paralysis, especially critical in sectors like healthcare and industrial automation. Jamming attacks' unpredictable nature and IoT devices' energy constraints complicate defence strategies. Addressing these challenges requires comprehensive approaches, focusing on advanced localisation methods and energy-efficient solutions. The evolution of jamming threats has spurred the development of more precise localisation techniques, where machine learning emerges as a promising tool for enhancing detection and accuracy.

This study investigates novel localisation methods in IoT networks against jamming interference, building upon Fuzzy Logic-based Intrusion Detection Systems (FLIDS). The research integrates FLIDS for initial jamming detection, forming a base for further localisation strategies. These strategies are evaluated using metrics like Euclidean error distance and network layer metrics such as Retransmissions and Expected Transmission Count (ETX). Advanced machine learning models, including RNN-B-LSTM, TCN, RNN and CNN Models, are incorporated for data handling capabilities and real-time processing suitability in IoT environments. The models are assessed through various metrics, with RNN-B-LSTM showing remarkable accuracy, making it suitable for precise localisation in jamming scenarios.

In conclusion, the RNN-B-LSTM model stands out for accurate jammer localisation in IoT networks, but the model selection should consider the specific IoT environment's demands and constraints. This study demonstrates machine learning's potential in counteracting jamming attacks and guides future research in technology integration for IoT network security. The novelty of this approach lies in integrating FLIDS [1], [2] for initial attack detection and employing advanced machine-learning models for precise jamming Source localisation. The empirical validation of these methods using simulations in Contiki OS and the Cooja tool with TelosB sensor nodes offers valuable insights for practical implementation in real-world scenarios.

The novelty of our approach lies in the seamless integration of a Fuzzy Logic-based Intrusion Detection System (FLIDS) with advanced machine learning models such as RNN-B-LSTM, TCN Model, RNN and CNN. This integration is tailored explicitly for efficient jamming attack detection and accurate localisation within IoT networks. Diverging from conventional detection methods, our strategy excels in recognising the presence of threats and precisely determining their origins. By leveraging the unique capabilities of FLIDS and these cutting-edge models, our approach sets a new benchmark in IoT network security, enabling both effective detection and targeted countermeasures against jamming attacks. The key contributions of this paper are summarised below:

1) **Novel and accurate ML localisation Methods for IoT**

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 739578, the ADROIT6G project of the SNS-JU under Grant Agreement No. 101095363, and the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

Networks: Examination of the efficacy of innovative ML localisation techniques in IoT networks facing jamming interference.
2) **Advanced Machine Learning Models:** Incorporation of RNN-B-LSTM, TCN Model, and RNN Model for handling complex data patterns and real-time processing in IoT environments.
3) **Highlighting RNN-B-LSTM Model's Superiority:** Demonstration of the RNN-B-LSTM model's effectiveness in jammer localisation through its high accuracy and reliability.
4) **Foundational Framework for Future Research:** Laying the groundwork for future exploration in integrating machine learning and IoT network security, particularly against jamming attacks.
5) **Offering a comprehensive methodology:** for detecting and localising Jammers security breaches through the utilisation of FLIDS (Fast Local Intrusion Detection System) technology.
6) **Network Metrics:** Our approach focuses on network metrics.
7) **Utilization of Multivariate Regression:** This study innovatively applies the Multivariate Regression technique [1] to enhance the accuracy of localization in IoT networks.

The rest of the paper is structured as follows. Section II provides related work on the ML localisation approaches and the background information on the localisation, jamming attacks, FLIDS and the investigated ML approaches. The problem description is elaborated in Section III. Next, the methodology used for our investigation is shown in Section IV along with the assumptions, evaluation metrics used, hyperparameters optimisation, system emulation setup and the localisation algorithm used to create the input features and send them as one-time series to the ML models for prediction of the x and y values of the jammer. The investigated approaches' results and efficiency are examined, evaluated and compared in Section V. Finally, Section VI includes concluding remarks and our future directions.

II. Related Work and Background Knowledge

A. Related Work

This section briefly reviews the open literature approaches related to approaches that utilise ML for localisation. Also, we compare the investigated approaches with the paper-examined approach.

Feng et al. [3] introduced a deep transfer learning method for disturbance localisation in power systems, employing Principal Component Analysis (PCA) and Smooth Pseudo Wigner-Ville Distribution (SPWVD). This method, notable for its two-stage approach, achieved up to 100% accuracy, significantly enhancing accuracy and efficiency in challenging scenarios, including those with noise and topology changes. Continuing with Abdallah et al. [4], they developed a method integrating received signal strength (RSS) from cellular towers with a Weighted K-nearest neighbour (WKNN) algorithm and a multi-layer neural network. This integration resulted in a mean localisation error of 5.9 meters and 5.1 meters in two urban environments and 8.7 meters in a rural setting, representing improvements of 41%, 45%, and 16%, respectively, over the WKNN-only approach. Next, Singh et al. [5] proposed SVR-based methods for predicting Average localisation Error (ALE) in wireless sensor networks. Among the methods evaluated, R-SVR outperformed others with a high correlation coefficient of 0.82 and a low RMSE of 0.147m, indicating superior prediction accuracy. Additionally, Yean et al. [6] employed Random Forest models for indoor localisation using Wi-Fi RSSI data. The methodology achieved high accuracy rates of 88.21% and 86.34% for the N4 and UJI datasets, respectively. Further, by reducing features to only 20% using PCA, the models maintained around 80% accuracy.

In their study, Berz et al. [7] evaluated. Artificial Neural Network (ANN) and Support Vector Regression (SVR) models for localising stationary objects in indoor environments. The ANN model consistently outperformed the SVR model, showing a 31% better performance on average. The localisation error for the ANN model ranged between 9 and 29 cm in various test scenarios. Also, Phillips and colleagues [8] introduced the Acoustic Landmark Locator (ALL) using acoustic sensing and neural networks for indoor localisation. The system demonstrated its ability to successfully determine a person's location in a realistic indoor environment, leveraging the unique acoustic characteristics of indoor spaces. Even more, Masoudinejad et al. [9] explored machine-learning techniques for indoor localisation in warehouses using IoT data. The Random Forest classifier, the most accurate model, improved its accuracy from 65% to 85% with the inclusion of reference values, emphasising its utility in complex warehouse environments.

Aqeel et al. [10] presented a study on node localisation in LoRaWAN environments, especially under sandstorm conditions. They employed SVR and GPR, showing that the multiple regression model performs well even in signal degradation, with an average RMSE for the ANN and SVR models being 20.6 cm and 16.3 cm, respectively. Finally, Islam and co-authors [11] investigated LoRa technology for localisation using machine learning models. Their approach, which combined range estimation and trilateration, improved localisation accuracy by about 10% for ranging and 26.58% for trilateration-based localisation compared to using RSSI alone. The method achieved an average distance error of 43.97 meters, significantly improving over traditional machine-learning techniques.

Table I compares various localisation methods in IoT networks. Our approach stands out as it utilises a combination of machine learning techniques, including RNN-B-LSTM, RNN, TCN, and CNN, tailored explicitly for IoT networks. This approach achieves an average distance error of 0.094 meters and supports jammer identification in the network layer (RPL). Compared to other methods in the table, it offers superior performance in terms of accuracy and the ability to detect and locate jammers, making it a robust choice for localisation in IoT networks. Finally, it can provide in one evaluation the x and y at the same time as numerical values (Multivariate Regression).

[1]Multivariate Regression is particularly effective in this context as it allows for the simultaneous analysis of multiple predictor variables. This is essential in IoT environments where factors such as signal strength, noise, and interference influence localisation accuracy. By integrating multiple variables, Multivariate Regression provides a more comprehensive distinction in the understanding of the underlying patterns and relationships, leading to more accurate and reliable predictions than those derived from univariate methods.

TABLE I: Comparison of ML Localisation Methods in IoT Networks

Ref.	**Approach**	**Key Features**	**Performance Metrics**	**Network Layer Metrics**	**Jammer Locali-sation**	**Avg. Dist. Error (m) <0.30**	**Multivariate Regres-sion (X,Y)**	**Most Accu-rate ML**
Feng et al. [3]	Deep Transfer Learning (DTL)	PCA, SPWVD, Two-stage approach for system and area-level localisations	Up to 100% accuracy in challenging conditions	No	No	Yes (< 0.30)	No	DTL
Abdallah et al. [4]	RSS with WKNN and Neural Network	Integration of WKNN's clustering and neural network position estimation	Mean localisation error: 5.9m, 5.1m (urban), 8.7m (rural)	No	No	No	No	Neural Network
Singh et al. [5]	SVR Methods (S-SVR, Z-SVR, R-SVR)	Features from modified Cuckoo Search simulations	R-SVR: Correlation coefficient 0.82, RMSE 0.147m	No	No	Yes (0.147)	No	R-SVR
Yean et al. [6]	Random Forest with Wi-Fi RSSI Data	Feature selection, Hyper-parameter tuning, PCA	Accuracy: 88.21% (N4 dataset), 86.34% (UJI dataset)	No	No	No	No	Random Forest
Berz et al. [7]	ANN and SVR for Object Localization	Multi-sensor system, CV subsystem for RFID localization	ANN outperforms SVR with 9-29 cm localization error	No	No	Yes (0.09 - 0.29)	No	ANN
Phillips et al. [8]	Acoustic Landmark Locator (ALL)	Neural network algorithm, Acoustic sensing	Effective indoor localization with landmark accuracy	No	No	Yes (< 0.30)	No	Neural Network
Masoudinejad et al. [9]	Machine Learning in Warehouse Localization	SVM, DT, RF, KNN, Gaussian Naive Bayes, IoT data	Random Forest most accurate: 65%-85% with reference values	No	No	No	No	Random Forest
Aqeel et al. [10]	SVR and GPR in LoRaWAN	RSSI data processing, Multiple regression model	Average RMSE: 20.6 cm (ANN), 16.3 cm (SVR)	No	No	Yes (0.206, 0.163)	No	SVR
Islam et al. [11]	LoRa Technology with ML Models	RF, kNN, SVM, GB, MLP, RSSI, SF, SNR	Average distance error: 43.97 meters	No	No	No	No	MLP
Our Approach	FLIDS with Advanced ML Models	Integration of FLIDS for jamming detection, RNN-B-LSTM for precise localisation along with the TCN, RNN and CNN	MSE: 0.006, MAE: 0.058, RMSE: 0.081, R-squared: 0.999, Median AE: 0.040, Avg Dist Error: 0.094m	Yes	Yes	Yes (0.094)	Yes	RNN-B-LSTM

B. Background Knowledge

This section presents background information that we used in our approach for the localisation algorithms. More specifically, we examine the localisation types and show the centroid algorithm. The types of jamming attacks, the FLIDS detection algorithm that we used, and a description of the investigated ml approaches.

1) Types of Localisation algorithms: The area of localisation algorithms can be broadly categorised into Range-based algorithms and Range-free algorithms. Range-based algorithms (i.e., Time of Arrival (ToA), Time Difference of Arrival (TDoA), Angle of Arrival (AoA), and Received Signal Strength (RSS)) they employ precise distance or angle measurements to determine location. These techniques, as described in [12]–[18], frequently necessitate the utilisation of advanced and expensive equipment such as directional antennas for measuring distances [19]–[21]. Conversely, Range-free algorithms rely on the connectivity data between nodes and use protocols that do not depend on radio signal strength measurements (i.e., the Centroid Algorithm, Approximate Point In Triangulation (APIT), DV-Hop algorithm, and the Amorphous Positioning Algorithm) by determining a node's location based on its communication range with other nodes. If two nodes can communicate, they are likely within their maximum transmission range of each other [21]–[23].

The Range-based methods are known for their precision. Range-free approaches are gaining attention as more economical solutions, mainly when high precision is not paramount [24]. However, our approach is in the second category, "Range-free" algorithms, and achieves high precision.

2) Centroid Localisation Technique: The Centroid localisation Technique is analysed in this research because the investigated approach uses the results of this technique. So, the Centroid approach is utilised to localise network jammers by analysing the positions of nodes affected by jamming. This method calculates the jammer's location by averaging the coordinates of all jammed nodes [25]. Centroid's robustness to radio propagation uncertainties is countered by its sensitivity to the spatial distribution of jammed nodes, where biased estimations can occur due to uneven distribution. The accuracy of the centroid improves with a uniform distribution of jammed nodes in a dense network.

Given N jammed nodes, represented as $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)\}$, the algorithm for estimating the jammer's location is shown in Algorithm 1 and the equation used by the algorithm is shown in the Equation 1.

We excluded the Centroid approach as a competitive approach from our investigation for two reasons. Firstly, it is not a machine-learning approach. Secondly, the centroid estimates are unsuitable for our approach due to the negative R-squared values for both X (-0.347) and Y (-0.288), indicating that the models could be better fitted compared to a simple average. Moreover, the substantial Mean Squared Error values for X (2698.02) and Y (2579.90), as well as the Root Mean Squared Error values for X (51.94) and Y (50.79), suggest considerable average forecast inaccuracies. So, these estimations cannot be used for precise localisation. However, we use the centroid output for x and y as features of our ML models (as shown in Section IV-C).

Algorithm 1 Centroid Algorithm

```
 1: Initialize sum of x-coordinates, $S_X = 0$
 2: Initialize sum of y-coordinates, $S_Y = 0$
 3: Initialize number of jammed nodes, $N = 0$
 4: for each jammed node $(X_i, Y_i)$ do
 5:     $S_X \leftarrow S_X + X_i$
 6:     $S_Y \leftarrow S_Y + Y_i$
 7:     $N \leftarrow N + 1$
 8: end for
 9: Calculate centroid coordinates:
10: $\widehat{X}_{\text{jammer}} \leftarrow \frac{S_X}{N}$
11: $\widehat{Y}_{\text{jammer}} \leftarrow \frac{S_Y}{N}$
12: return $(\widehat{X}_{\text{jammer}}, \widehat{Y}_{\text{jammer}})$
```

$$\left(\widehat{X}_{\text{jammer}}, \widehat{Y}_{\text{jammer}}\right) = \left(\frac{\sum_{i=1}^{N} X_i}{N}, \frac{\sum_{i=1}^{N} Y_i}{N}\right) \qquad (1)$$

3) Types of Jamming Attacks: The two primary categories of Jammer types are Proactive, and Reactive [26], [27].

The Proactive jammers consistently initiate network disturbances and can be categorised into three distinct forms. First, the **Constant Jammer** which emits uninterrupted and unpredictable signals, disrupting network communications without following MAC rules. Next, the **Deceptive Jammer** employs a strategy of transmitting packets that appear genuine, hence complicating detection efforts. However, it emits data packets that cannot be identified, albeit at the cost of reduced energy efficiency. Finally, the **Random Jammer** switches between periods of inactivity and activity, perhaps mimicking the behaviour of the first two varieties when it is active. It has a higher energy efficiency but a lower level of constant effectiveness.

Continuing with the Reactive jammers, these jammers monitor the activities on the channel and emit a random signal to disrupt ongoing transmissions when activity is detected. Monitoring a channel requires considerably less power than proactive jamming. The **Reactive jammers** exemplify this category of jammers, exhibiting characteristics such as being challenging to detect, intricate to construct, less energy-efficient, and limited to operating on a single channel [28].

4) FLIDS (Fuzzy Logic-based Intrusion Detection System): FLIDS, or the Fuzzy Logic-based Intrusion Detection System, is an intelligent and adaptive technique designed for detecting various types of jamming attacks in Internet of Things (IoT) networks. It operates in a distributed manner, meaning detection is performed locally using information sourced or calculated at the node level. The system utilises a combination of metrics such as "Expected Transmission Count (ETX) & Retransmissions" and "Packets Drop per Terminal (PDPT) & Retransmissions" as inputs into a Fuzzy inference system. This results in generating a Jamming Index (JI)[2], which is key to identifying jamming attacks and quantifying the degree of jamming impact on a node. Initially, the algorithm gathers crisp values of ETX, PDPT, PDR, and Retransmissions from network nodes. These values are then fuzzified, preparing them for processing through fuzzy logic. The essence of the algorithm is its application of a specific fuzzy rule base to these inputs. For example, a rule might infer that a combination of 'Low' ETX and 'High' Retransmissions suggests a 'Medium' level of jamming, indicated by the JI. After applying these rules, the algorithm produces a fuzzy output, subsequently defuzzified to provide a clear, quantifiable measure of jamming. This final output, the Jamming Indicator, is a crucial metric representing the likelihood or intensity of a jamming attack on a node. This process allows for effective identification and mitigation of potential jamming issues in the network. The approach has demonstrated high accuracy in recognising different types of jamming attacks across various situations, and it is characterised by low memory usage and fast execution times. This makes FLIDS a suitable and efficient choice for IoT environments where resources are often limited [28].

[2]The Jamming Index (JI) is a numerical value ranging from 0 to 1, used to quantify the level of jamming in a network. A JI close to 0 indicates no jamming, while a value approaching 1 signifies severe jamming. It is calculated using network metrics like Expected Transmission Count (ETX) and Retransmissions, processed through a Fuzzy Inference System (FIS), which makes it a suitable measure in scenarios where network conditions are uncertain or variable.

a) Explanation on Parameters used for Jamming Attack Detection: Existing systems for detecting jamming attacks have utilised a significant multitude of parameters [28]. These metrics are measured in multiple layers, including received or dropped packets, collisions or (re)transmissions, routing parameters such as hops and control packet numbers, energy consumption, and physical signal parameters. Our approach utilises ETX (Expected Transmission Count), Retransmissions, and PDPT (Packets Dropped per Terminal). These inputs are fed into a fuzzy inference system. The selection of these values is grounded in prior research [28], which shows that the amalgamation of "ETX & Retransmissions" and "PDPT & Retransmissions" yielded the highest level of accuracy. Additionally, both methods rely on distributed data and can be acquired and executed locally at nodes, which we find attractive. The parameters used are described briefly as follows: i) The Retransmissions are the number of repeated transmissions needed for a frame to be effectively relayed to the subsequent node; ii) The Packets Dropped per Terminal (PDPT) represent the proportion of received packets that did not pass the Cyclic Redundancy Check (CRC) conducted by the node, compared to the total number of packets received by the node within a specific time frame; and iii) The Expected Transmission Count (ETX) is a measure that indicates the anticipated number of transmissions needed to send and receive a packet over a wireless network successfully. It is important to note that the FLIDS end up using the "ETX & Retransmissions" metrics, where ETX is a parameter of Routing Protocol for Low-Power and Lossy Networks (RPL) [29], thus making this work one of the few, if not the only one considering such routing-level parameters for Jamming detection.

C. Investigated ML approaches

In this section, we show the ML approaches that we investigate in this examination. The examination approaches are the folloiwing:

1) **RNN-B-LSTM (Recurrent Neural Network - Bidirectional Long Short-Term Memory)**: This approach utilises a bidirectional LSTM structure within a recurrent neural network. It is designed to process sequences of data by maintaining internal memory states. The bidirectional architecture allows the network to have

both forward and backward information about the sequence at every point (as shown in Eq. 2).

$$h_t = f(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}) \tag{2}$$

where h_t is the hidden state at time t, x_t is the input at time t, and W and b are parameters of the model.

2) **TCN Model (Temporal Convolutional Network)**: The TCN model employs convolutional layers designed explicitly for sequence modelling tasks. It uses causal convolutions, ensuring that the output at time t is convolved only with elements from time t and earlier in the previous layer (as shown in Eq. 3).

$$y_t = \sum_{i=0}^{N-1} f(x_{t-i}) \cdot w_i \tag{3}$$

where y_t is the output at time t, N is the size of the kernel, x is the input, w is the weight, and f is a nonlinear function.

3) **RNN Model (Recurrent Neural Network)**: RNNs are a type of neural network where connections between nodes form a directed graph along a temporal sequence, enabling it to exhibit temporal dynamic behaviour. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs (as shown in Eq. 4).

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{(t-1)} + b) \tag{4}$$

where σ is the activation function, h_t is the hidden state at time t, x_t is the input, and W and b are the network weights and biases.

4) **CNN Model (Convolutional Neural Network)**: CNNs are primarily used in analysing visual imagery and are known for their ability to recognise image patterns. They employ a mathematical operation called convolution, a specialised linear operation. CNNs are used to detect and interpret complex features in data (as shown in Eq. 5).

$$Z = \sum_{i,j} (X * K)_{i,j} \tag{5}$$

where Z is the output, X is the input, K is the kernel, and $*$ denotes the convolution operation.

III. Problem Description

In IoT networks, jammers pose a significant threat as they can disrupt wireless communications, leading to network inefficiencies or complete failures. These attacks are particularly critical in healthcare and industrial automation sectors, where reliable data transmission is essential. Mitigating these threats or implementing targeted security measures effectively is easier with accurate localisation of jammers. Localising the jammer allows for precise countermeasures, such as adjusting network protocols or physically securing vulnerable areas. Effective localisation also aids in understanding attack patterns, improving overall network resilience and security. In IoT environments, where resources are often limited, precise jammer localisation is vital for maintaining uninterrupted operations and ensuring the integrity of critical data flows.

IV. Methodology

Starting with the methodology, we need to indicate that our examination focuses on the localisation of a jammer with the use of time series ML approaches that support multivariate regression after the identification of jamming attacks from FLIDS. Before we start with the localisation strategy, we need to indicate that we used the output of the FLIDS fuzzy logic system (Jamming Index) as an input parameter in our approach. Also, we need to suggest that the JI indicates the presence or level of jamming in a communication system [28]. Continuing, this section includes the assumptions, evaluation metrics used to assess the models, the process and evaluation of feature selection, the technique employed for optimising hyperparameters in our models, the system emulation and setup, and the localisation algorithm used to determine the accurate positions of the jammer.

A. Assumptions

Our investigation considers the following assumptions:

- Some sensors are expected to report data to the designated sink node, even with the jammer. The data to be reported includes the Jamming Index and their respective locations.
- In our investigation, the IoT sensors are not mobile, and they have predefined locations, and we know accurately these positions 100%.
- The communication network infrastructure is assumed to be operational and capable of transmitting data from sensors to the sink.
- Sensors are assumed to have the necessary hardware and software components to collect and transmit data accurately.
- The sink node is assumed to be able to receive, process, and store data from all sensors effectively.
- The project assumes the availability of power sources or batteries for the sensors to function continuously.
- Environmental conditions, such as weather or interference, are assumed to be within acceptable limits for sensor operation.

B. Evaluation Metrics Used

In this section, we explain the metrics that are used in our examination. These are the following [30]:

- **Mean Squared Error (MSE)** [31]: A metric that measures the average of the squares of the errors, essentially the average squared difference between the estimated values and the actual value. Formula: MSE $= \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$. A lower MSE indicates a closer fit to the data.
- **Mean Absolute Error (MAE)** [31]: Represents the average of the absolute differences between the predicted values and the observed actual outcomes. Formula: MAE $= \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$. The lower the MAE, the better the model's accuracy.
- **Root Mean Squared Error (RMSE)** [31]: The square root of the mean of the squared differences between prediction and actual observation. Formula: RMSE $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$. It is a measure of the magnitude of the prediction error.
- **R-squared** (R^2) [31]: Provides an indication of the goodness of fit of a set of predictions to the actual values. Formula: $R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$. It represents the proportion of variance explained by the model.
- **Median Absolute Error** [32]: This is the median of all absolute differences between the actual and the predicted values. Formula: Median Absolute Error $=$ median$(|Y_i - \hat{Y}_i|)$. It is less sensitive to outliers.

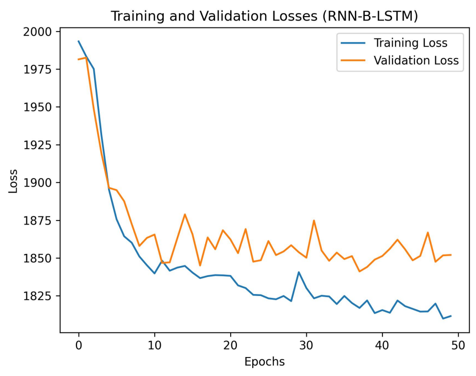


Fig. 1: RNN-B-LSTM prediction without the Centroid Approach Features

- **Explained Variance Score** [33]: Measures the proportion to which a model accounts for the variation of a dataset. Formula: Explained Variance $= 1 - \frac{\text{Var}(Y - \hat{Y})}{\text{Var}(Y)}$. A higher score indicates better model performance.
- **Average Distance Error** [30]: Reflects the average distance between the predicted and actual values. Formula: $\text{Average Distance Error} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(Y_{i,x} - \hat{Y}_{i,x})^2 + (Y_{i,y} - \hat{Y}_{i,y})^2}$. Lower values indicate higher precision.

C. Features selection

Feature selection is a critical step in machine learning that involves choosing the relevant input variables that contribute most significantly to the model's predictive power. In our approach, we start by preparing a dataset containing various attributes related to network nodes' communication parameters. From this dataset, we systematically exclude variables that are less likely to influence the location prediction (such as the ETX, Retransmissions, and Drop Packets) because they are related to the Fuzzy Index. When we started our evaluation, we used the fuzzy index, sensor x location, and sensor y location as features. We use each case as a one-time series. However, the RNN-B-LSTM model's performance shows notable prediction errors, as indicated by a Mean Squared Error of 1841.904 and a Root Mean Squared Error of 42.917. With a Mean Absolute Error of 36.842 and a Median Absolute Error of 37.989, the average prediction deviations are significant. The low R-squared (0.073) and Explained Variance Score (0.073) imply limited model effectiveness in explaining the data's variability. The Average Distance Error of 56.555 reflects the model's limited precision (as shown in Figure 1).

Thus, we introduced the calculation of two extra features, which will increase the model's accuracy. The two features are the Centroid Value of X and the Centroid Value of y. To examine the features, we execute multiple feature selection techniques, such as [34]:

- **Fisher Scores [35]:** This method evaluates features' discriminative power by calculating the variance ratio between different classes to the variance within each class. A higher Fisher score for a feature indicates a greater ability to differentiate between classes. The Fisher Score is given by the following equation:

$$F_j = \frac{\sum_{i=1}^{c} n_i(\mu_{ij} - \mu_j)^2}{\sum_{i=1}^{c} n_i\sigma_{ij}^2} \tag{6}$$

  where F_j is the Fisher Score for the j-th feature, c is the number of classes, n_i is the number of samples in the i-th class, μ_{ij} is the mean of the j-th feature in the i-th class, μ_j is the overall mean of the j-th feature, and σ_{ij}^2 is the variance of the j-th feature in the i-th class.
- **Mutual Information Scores [36]:** Mutual Information quantifies the shared information between a feature and the target variable. It is calculated by assessing how much knowing one of these variables reduces uncertainty about the other. Higher mutual information values suggest a stronger relationship between the feature and the target variable. It is calculated by the following equation:

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{7}$$

  where $I(X;Y)$ is the mutual information between feature X and target Y, with X, Y representing the feature and target as random variables, $p(x,y)$ being the joint probability distribution function of X and Y, and $p(x)$, $p(y)$ the marginal probability distribution functions of X and Y respectively.
- **Chi-Square Scores [37]:** The Chi-Square test measures the dependence between a feature and the target variable. It compares the observed occurrences of features against the expected occurrences if the feature and target were independent. Higher scores indicate a stronger association with the target variable. The Chi-Square score is calculated by the following equation:

$$\chi^2 = \sum_{i=1}^{n}\frac{(O_i - E_i)^2}{E_i} \tag{8}$$

  where χ^2 is the Chi-Square score, O_i is the observed frequency, E_i is the expected frequency under the assumption of independence, and n is the number of categories or classes.
- **Random Forest Feature Importances [38]:** In a Random Forest model, the importance of a feature is determined by how much it decreases the impurity in the decision trees. It's typically measured by the reduction in Gini impurity or mean decrease in impurity caused by the feature. Higher importance values indicate a more significant role in the model's predictions. The feature importance is calculated as:

$$FI = \sum_{t\in T} p(t)\cdot\Delta i(s,t) \tag{9}$$

  where FI is the feature importance, t represents each node in the set of trees T within the Random Forest model, $p(t)$ is the proportion of samples that reach node t, s denotes the feature being evaluated, and $\Delta i(s,t)$ is the decrease in impurity (e.g., Gini impurity) at node t attributed to feature s.

Finally, Figure 2 validates the significance of the selected features shown in Table II for machine learning training, as evidenced by their high scores across Fisher Score, Information Gain, Chi-Square, and Random Forest Importance metrics, indicating that they should be retained for effective model training.

D. Hyperparameters optimisation

Hyperparameters are the configuration settings used to structure machine learning models. Unlike model parameters learned during training, hyperparameters are set in advance

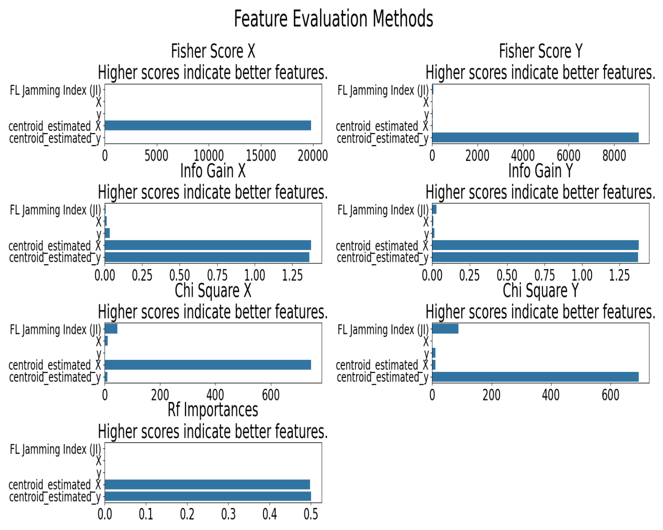


Fig. 2: Features Evaluation

TABLE II: Selected Features for ML Training

Type	Features
Input[1]	Jamming Index, Sensor_X, Sensor_y, Centroid_x_result, Centroid_y_result
Output[2]	Jammer_x_position, Jammer_y_position

[1] Input Numerical Selected Features
[2] Output Numerical Expected Features

and guide the training process. Their optimisation is crucial as they have a profound impact on the performance of the model. In our approach, we implement hyperparameter optimisation using the "RandomSearch algorithm", which is an exhaustive search method that randomly samples the hyperparameter space and evaluates sets of hyperparameters within predefined bounds. For the RNN model, hyperparameters such as the number of units in the RNN layer and the learning rate of the optimiser are tuned. The number of units determines the capacity of the network to learn from data, while the learning rate controls the speed at which the model learns during training. Similarly, for the CNN model, we tune hyperparameters like the number of filters and kernel size in the convolutional layers, along with the learning rate. The number of filters affects the model's ability to extract features from the input data, and the kernel size determines the extent of the receptive field over which the model perceives patterns. For example, in our examination, the optimal hyperparameters for the more accurate approach, which is the RNN-B-LSTM model, were determined as follows: i) First LSTM layer units: 192; ii) Dropout rate: 0.2; and iii) Learning rate: 0.006.

E. System Emulation Setup

The system emulation setup assesses a deceptive jamming attack in an Internet of Things (IoT) network environment, utilising the Contiki operating system and the Cooja simulator for experimental validation. The experiments leverage the JamLab suite, a Contiki-based library that facilitates reproducible interference testing by simulating a jammer that disrupts network communications with seemingly legitimate signal packets. In the simulation setup (shown in figure 3[3]), a grid comprising 25 nodes, including a centrally located Sink node, is established across a 160 by 160-meter field. The Sink node is fixed at the centre for all scenarios. Each node in the grid is positioned equidistantly at 40 meters from its neighbours, emulating a network of TelosB nodes with individual transmission and interference ranges of 50 and 70 meters, respectively. The nodes are configured to send a 48-byte data packet at regular intervals of 10 seconds, thereby maintaining a consistent data transmission schedule within the network.

[3]This visualisation where the jammer is at the 12th position is crucial for understanding the jamming's spatial effect on the network's communication capabilities, offering empirical evidence to evaluate the IoT network's defence mechanisms against jamming attacks.

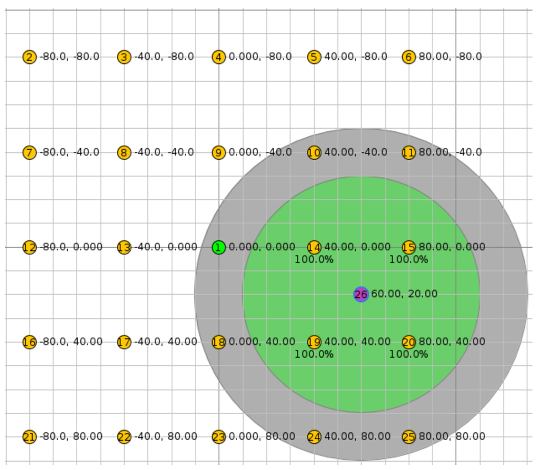


Fig. 3: The architecture of the emulation system.

Next, to analyse the robustness of the network under jamming conditions, one hundred randomly generated scenarios are considered based on the jammer's location relative to the grid. Note that the Sink node is always at the centre. For each scenario, 16 different jammer positions are tested and placed in the grid in random order as cases. The simulation data extracted from the Cooja Simulator is processed and analysed through Python and MATLAB for the Jamming Index and the attack identification using the FLIDS, providing quantitative insights into the jamming impact on network communication.

Algorithm 2 Jammer Position Identification Using Fuzzy Logic and ML

```
1:  Input: Sensor data, Fuzzy index
2:  Output: Predicted Jammer position (x, y)
3:  for each case of one jammer position do
4:      Gather Fuzzy index and sensor location from responsive sensors as a time series
5:      Initialize list AffectedSensors
6:      for each sensor do
7:          if Fuzzy index of sensor $> 0.5$ then
8:              Add sensor to AffectedSensors
9:          end if
10:     end for
11:     Initialize TotalX, TotalY, Count to 0
12:     for each sensor in AffectedSensors do
13:         TotalX += x-coordinate of sensor
14:         TotalY += y-coordinate of sensor
15:         Count += 1
16:     end for
17:     Calculate centroid X: CentroidX = TotalX / Count
18:     Calculate centroid Y: CentroidY = TotalY / Count
19:     Assign CentroidX and CentroidY to all records of the case
20:     Use ML model to predict jammer position X and Y based on the time series
21: end for
```

F. Sink Hole/Gateway Localisation Algorithm

This section provides the role of the Gateway in our investigation. The Gateway gathers the information from the sensors and executes the localisation decision. Moreover, it processes the gathered information and executes the localisation algorithm shown at Algorithm 2. Note that we use time series in our case, and each case is a one-time series to calculate the multivariate regression values of x and y as outputs.

V. Results and Performance Evaluation

This section examines, evaluates, and compares the efficiency of the examined RNN-B-LSTM, TCN, RNN and CNN ML approaches. Each approach is assessed using the following figures: i) Training vs validation Loss figure, ii) Actual metrics vs predicted metrics figure, and iii) 3D Actual vs predicted Values Figure. Note that we use the 3D representation of the "3D Actual vs predicted Values Figure" to show each ML approach's prediction errors clearly.

a) RNN-B-LSTM: Starting with the RNN-B-LSTM, as shown in Figures 4, the RNN-B-LSTM model demonstrates exceptional performance in predicting location coordinates. The training and validation loss curves indicate a rapid convergence to a low-error state, showcasing the model's ability to learn from the data effectively. Scatter plots of the actual versus predicted values for both X and Y coordinates show a tight correlation around the ideal diagonal, underscoring the model's precise predictive accuracy. The 3D scatter plot further reinforces this, with the predicted values closely clustering around the actual data points in the three-dimensional space, which reflects the model's robustness in handling complex spatial patterns. Overall, the visualisations confirm the RNN-B-LSTM's high reliability and accuracy in localisation tasks within the given context.

b) TCN: Continuing with the TCN, as shown in Figures 5, the training and validation losses decrease sharply and plateau early, suggesting quick model convergence. The scatter plots for both X and Y dimensions reveal a high degree of correlation between the actual and predicted values, with the data points clustering tightly along the line of perfect prediction. However, with a medium accuracy. The 3D scatter plot visualises the model's predictive performance, where the predicted points are close but around to the actual ones, indicating a medium capability of the TCN model to accurately capture the spatial relationships in the data.

c) RNN: The RNN, as shown in Figures 6, the line chart illustrates the training and validation loss over multiple epochs, showing both decreasing sharply before plateauing, indicative of the model's learning and generalisation capability. The adjacent scatter plots further illuminate the model's predictive accuracy by juxtaposing the actual and predicted values for X and Y dimensions, with points clustered very close to the diagonal line, denoting high precision. Finally, the 3D scatter plot offers a visual comparison of actual and predicted values in a three-dimensional space, where the proximity of the red (predicted) and blue (actual) points suggests the model's high performance in capturing the complex patterns in the data across multiple dimensions.

d) CNN: The CNN, as shown in Figures 7, illustrates the training history and prediction accuracy of a Convolutional Neural Network (CNN) model. The training and validation loss graph exhibits a significant reduction in loss during the initial epochs, followed by a stable convergence, indicative of effective learning and generalisation. The scatter plots for the actual versus predicted X and Y values show medium accuracy, with points clustering close to the ideal prediction line. The 3D scatter plot reinforced this accuracy across multiple dimensions, displaying a not-too-close overlap between the actual (blue) and predicted (red) values, suggesting the CNN model's adeptness in capturing the complexity of the data. Collectively, these figures point towards a well-trained CNN model with medium predictive performance.

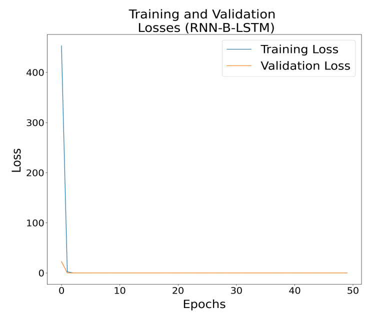
A. Overall Comparison

As shown in Table III, the RNN-B-LSTM model demonstrated exceptional precision, as reflected by its Mean Squared Error (MSE) of 0.006 and Mean Absolute Error (MAE) of 0.058. The Root Mean Squared Error (RMSE) of 0.081 further underscores its accuracy, complemented by an R-squared value close to unity at 0.999, indicating an almost perfect fit. The Median Absolute Error (Median AE) was notably minimal at 0.040, and the Explained Variance Score was equally high, matching the R-squared value. The model's Average Distance Error (Avg Dist Error) in meters stood at a mere 0.094 m, suggesting its robustness in localisation tasks. Additionally, the Temporal Convolutional Network (TCN) Model, in contrast, showed an MSE of 42.180 and an MAE of 5.257, indicating a less precise but still notable performance. Its RMSE reached 6.494, with an R-squared value of 0.979, which, while lower than the RNN-B-LSTM, still demonstrated substantial model accuracy. The Median AE at 4.649 and an Explained Variance Score of 0.989 suggested good predictive capabilities, though with a higher average Dist Error of 8.234 m, indicating a greater deviation from actual values than the RNN-B-LSTM model. Similarly, the RNN Model presented an MSE of 0.325, an MAE of 0.096, and an RMSE of 0.570, all of which denote a performance reduction when compared to the RNN-B-LSTM. However, its accuracy remained high, with an R-squared value of 0.999 and a Median AE of 0.054. The Explained Variance Score was consistent with the R-squared value, and the Avg Dist Error was 0.152 m, which was larger than the RNN-B-LSTM but still indicated a respectable level of precision. Moreover, the Convolutional Neural Network (CNN) Model recorded an MSE of 21.928, an MAE of 2.994, and an RMSE of 4.682, reflecting a moderate accuracy level. Its R-squared value of 0.989 and Explained Variance Score of 0.989 suggested a reliable predictive ability, albeit less than the models above. The CNN Model's Median AE was 1.864, and the average distance error was 4.581 m, marking it as less accurate regarding average distance but still adequate for specific localisation applications.

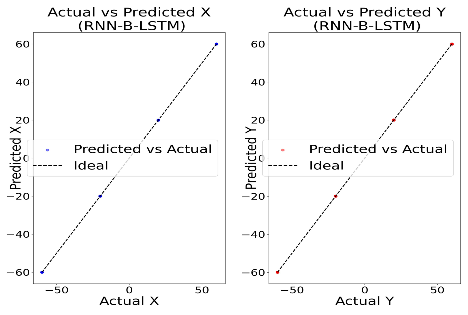
Overall, each model's performance metrics exhibit distinctive strengths and weaknesses, with the RNN-B-LSTM model demonstrating superior precision and the CNN Model balancing accuracy and computational efficiency. This comprehensive performance metric assessment allows for an informed selection of the most suitable model based on the specific requirements of the localisation task at hand.

VI. Conclusions and Future Work

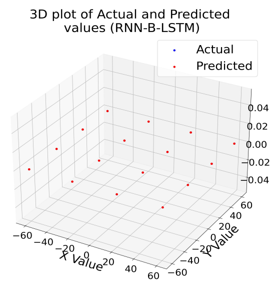
The study presented here is a significant stride towards fortifying the Internet of Things (IoT) against the pervasive



(a) Training Vs Validation Loss

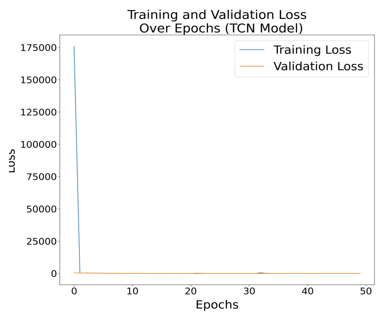


(b) Actual Vs Predicted

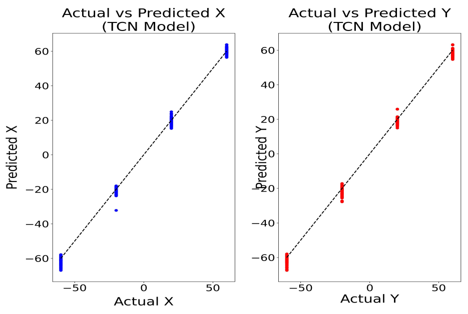


(c) 3D Actual Vs Predicted Values

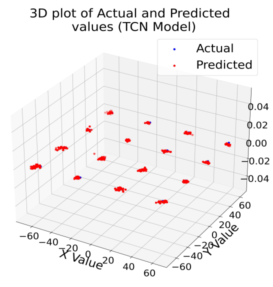
Fig. 4: RNN-B-LSTM Examinations



(a) Training Vs Validation Loss

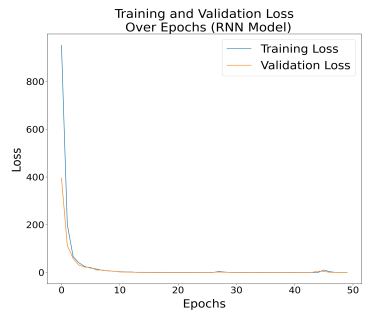


(b) Actual Vs Predicted

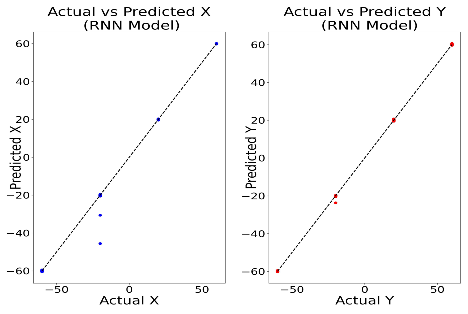


(c) 3D Actual Vs Predicted Values

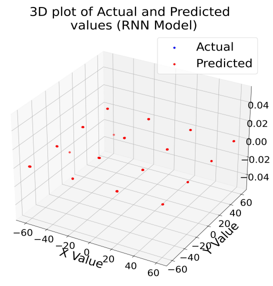
Fig. 5: TCN Examinations



(a) Training Vs Validation Loss

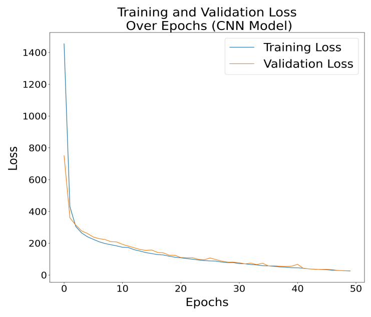


(b) Actual Vs Predicted

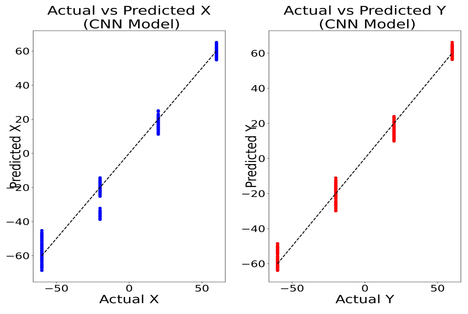


(c) 3D Actual Vs Predicted Values

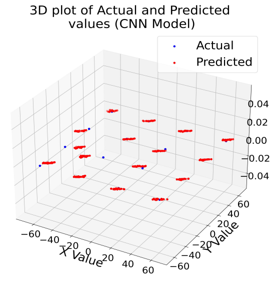
Fig. 6: RNN Examinations



(a) Training Vs Validation Loss



(b) Actual Vs Predicted



(c) 3D Actual Vs Predicted Values

Fig. 7: CNN Examinations

TABLE III: Performance Metrics of ML Models

Model	**MSE**	**MAE**	**RMSE**	**R-Squared**	**Median AE**	**Explained Variance**	**Average Distance Error (ADE) (in m)**
RNN-B-LSTM	0.006	0.058	0.081	0.999	0.040	0.999	0.0940
TCN Model	42.180	5.257	6.494	0.979	4.649	0.989	8.234
RNN Model	0.325	0.096	0.570	0.999	0.054	0.999	0.152
CNN Model	21.928	2.994	4.682	0.989	1.864	0.989	4.581

threat of jamming attacks, integrating advanced machine learning models with a Fuzzy Logic-based Intrusion Detection System (FLIDS) for detection and precise localisation of different threats. Our findings underscore the RNN-B-LSTM model's superior accuracy in jammer localisation within IoT networks, showcasing its potential in environments where precise localisation is critical, such as healthcare and industrial automation. Its low MSE and high R-squared value highlight the model's outstanding performance, making it a reliable choice for critical IoT applications.

Future work should focus on refining the TCN and RNN models to improve their localisation accuracy, exploring hybrid models that could combine the strengths of RNNs and CNNs, and further reducing the average distance error for all models to bolster precision. Moreover, continuous advancements in machine learning algorithms and their application in IoT will be essential in staying ahead of increasingly sophisticated jamming techniques. Moreover, in future research, it would be valuable to explore the implications of introducing uncertainty into the node locations or considering scenarios where one of the nodes possesses highly erroneous position information, as this would shed light on the robustness and reliability of the proposed system under real-world conditions.

References

[1] M. Savva, I. Ioannou, and V. Vassiliou, "Fuzzy-logic based ids for detecting jamming attacks in wireless mesh iot networks," in *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2022, pp. 54–63.

[2] ——, "Performance evaluation of a fuzzy logic-based ids (flids) technique for the detection of different types of jamming attacks in iot networks," in *2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2023, pp. 93–100.

[3] S. Feng, J. Chen, Y. Ye, X. Wu, H. Cui, Y. Tang, and J. Lei, "A two-stage deep transfer learning for localisation of forced oscillations disturbance source," *International Journal of Electrical Power and Energy Systems*, vol. 135, no. February 2021, 2022.

[4] A. A. Abdallah, S. S. Saab, and Z. M. Kassas, "A machine learning approach for localization in cellular environments," *2018 IEEE/ION Position, Location and Navigation Symposium, PLANS 2018 - Proceedings*, pp. 1223–1227, 2018.

[5] A. Singh, V. Kotiyal, S. Sharma, J. Nagar, and C. C. Lee, "A Machine Learning Approach to Predict the Average Localization Error with Applications to Wireless Sensor Networks," *IEEE Access*, vol. 8, pp. 208 253–208 263, 2020.

[6] S. Yean, B. S. Lee, and H. L. Oh, "Feature Engineering for Grid-based Multi-Floor Indoor Localisation using Machine Learning," *2020 International Conference on Intelligent Data Science Technologies and Applications, IDSTA 2020*, pp. 142–148, 2020.

[7] E. L. Berz, D. A. Tesch, and F. P. Hessel, "Machine-learning-based system for multi-sensor 3D localisation of stationary objects," *IET Cyber-Physical Systems: Theory & Applications*, vol. 3, no. 2, pp. 81–88, 2018.

[8] L. Phillips, C. B. Porter, N. Kottege, M. D'Souza, and M. Ros, "Machine learning based acoustic sensing for indoor room localisation using mobile phones," *Proceedings of the International Conference on Sensing Technology, ICST*, vol. 2016-March, pp. 456–460, 2016.

[9] M. Masoudinejad, A. K. Ramachandran Venkatapathy, D. Tondorf, D. Heinrich, R. Falkenberg, and M. Buschhoff, "Machine learning based indoor localisation using environmental data in phynetLab warehouse," *Smart SysTech 2018 - European Conference on Smart Objects, Systems and Technologies*, pp. 66–73, 2018.

[10] I. Aqeel, E. Iorkyase, H. Zangoti, C. Tachtatzis, R. Atkinson, and I. Andonovic, "LoRaWAN-implemented node localisation based on received signal strength indicator," *IET Wireless Sensor Systems*, vol. 13, no. 4, pp. 117–132, 2023.

[11] K. Z. Islam, D. Murray, D. Diepeveen, M. G. Jones, and F. Sohel, "Machine learning-based LoRa localisation using multiple received signal features," *IET Wireless Sensor Systems*, vol. 13, no. 4, pp. 133–150, 2023.

[12] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 32–43.

[13] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost Outdoor Localization for Very Small Devices," *IEEE personal communications*, vol. 7, no. 5, pp. 28–34, 2000.

[14] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free Localization Schemes for Large Scale Sensor Networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 81–95.

[15] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*. John Wiley & Sons, 2010.

[16] A. Kumar, N. Chand, V. Kumar, and V. Kumar, "Range Free Localization Schemes for Wireless Sensor Networks," *International journal of Computer Networks & Communications*, vol. 3, no. 6, p. 115, 2011.

[17] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2007.

[18] S.-Y. Kim and O.-H. Kwon, "Location Estimation Based on Edge Weights in Wireless Sensor Networks," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 30, no. 10A, pp. 938–948, 2005.

[19] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a Global Coordinate System from Local Information on an ad hoc Sensor Network," in *Information processing in sensor networks*. Springer, 2003, pp. 333–348.

[20] L. Lazos and R. Poovendran, "SeRLoc: Secure Range-independent Localization for Wireless Sensor Networks," in *Proceedings of the 3rd ACM workshop on Wireless security*, 2004, pp. 21–30.

[21] A. Ademuwagun, V. Fabio *et al.*, "Reach Centroid Localization Algorithm," *Wireless Sensor Network*, vol. 9, no. 02, p. 87, 2017.

[22] C.-Y. Chong and S. P. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[23] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in Wireless ad hoc Sensor Networks," *IEEE Transactions on computers*, vol. 52, no. 6, pp. 753–763, 2003.

[24] L. Pang, X. Chen, Z. Xue, and R. Khatoun, "A Novel Range-free Jammer Localization Solution in Wireless Network by Using PSO Algorithm," in *International Conference of Pioneering Computer Scientists, Engineers and Educators*. Springer, 2017, pp. 198–211.

[25] H. Liu, X. Wenyuan, Y. Chen, and Z. Liu, "Localizing Jammers in Wireless Networks," in *2009 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2009, pp. 1–6.

[26] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, pp. 46–57.

[27] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A Survey on Jamming Attacks and Countermeasures in WSNs," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, 2009.

[28] M. Savva, I. Ioannou, and V. Vassiliou, "Fuzzy-logic based ids for detecting jamming attacks in wireless mesh iot networks," in *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*. IEEE, 2022, pp. 54–63.

[29] H. Kharrufa, H. A. A. Al-Kashoash, and A. H. Kemp, "Rpl-based routing protocols in iot applications: A review," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 5952–5967, 2019.

[30] "Top 10 Machine Learning Evaluation Metrics for Regression - Implemented in R - Machine Learning, R programming." [Online]. Available: https://appsilon.com/machine-learning-evaluation-metrics-regression/

[31] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

[32] A. Botchkarev, "A new typology design of performance metrics to measure errors in machine learning regression algorithms," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 14, pp. 045–076, 2019.

[33] V. Plevris, G. Solorzano, N. P. Bakas, and M. E. A. Ben Seghier, "Investigation of performance metrics in regression analysis and machine learning-based prediction models," in *8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2022)*. European Community on Computational Methods in Applied Sciences, 2022.

[34] A. Gupta, "Feature Selection Techniques in Machine Learning (Updated 2023)," 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/#h-supervised-techniques

[35] Q. Gu, Z. Li, and J. Han, "Generalized fisher score for feature selection," *arXiv preprint arXiv:1202.3725*, 2012.

[36] G. Doquire and M. Verleysen, "Mutual information-based feature selection for multilabel classification," *Neurocomputing*, vol. 122, pp. 148–155, 2013.

[37] X. Jin, A. Xu, R. Bie, and P. Guo, "Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles," in *Data Mining for Biomedical Applications: PAKDD 2006 Workshop, BioDM 2006, Singapore, April 9, 2006. Proceedings*. Springer, 2006, pp. 106–115.

[38] D. Dutta, D. Paul, and P. Ghosh, "Analysing feature importances for diabetes prediction using machine learning," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2018, pp. 924–928.