Journal of Information Systems Engineering and Management
2025, 10 (32s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Anomaly Detection in Edge Computing using Deep Fuzzy Hypersphere Neural Network Learning Model on NSL-KDD Dataset

Sonali Jadhav[1][0009-0001-3731-3823] and Dr. Arun Kulkarni [2][0009-0008-7699-8272]

[1] Thadomal Shahani Engineering College, Mumbai University, Mumbai, India
[2] Thadomal Shahani Engineering College, Mumbai University, Mumbai, India
sonali.jadhav@thadomal.org
kkkarun@yahoo.com

ARTICLE INFO

Received: 24 Dec 2024

Revised: 12 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

IoT devices have been extensively utilized on numerous smart applications such as smart city, healthcare, and Industry. Since IoT devices possess tiny computing power and not capable to compute large volumes of data, in spite of the advantages of IoT, it also possesses inherent drawbacks like latency, bandwidth limitation, reliability concerns, and security risks. Edge computing counteracts these drawbacks by processing the data locally and implemented for processing this much huge sensors data on cloud. The Edge will process the data closer to where it is created so that processing may be accelerated and latency can be reduced, again in Edge Computing a variety of irregularities in data generation are generated by the increasing heterogeneity and complexity of edge devices due to their limitations. Anomaly detection is a crucial task in edge computing systems, where identifying unusual or deviant patterns of data is essential to ensuring system security and reliability. An original Deep Fuzzy Hypersphere neural network learning model (DFHNNLM) is proposed in this paper for effective anomaly detection in edge computing tasks. The proposed method outperforms current state-of-the-art for anomaly detection with existing deep learning techniques. Proposed model is suitable for any anomaly dataset like ECG5000, NSL-KDD. According to the experimental results, the DFHNNLM outperforms both deep learning and conventional machine learning methods in anomaly detection, achieving improvements in F1-score, accuracy, precision, and recall.

Keywords: Anomaly detection, Deep Neural Network, Edge Computing, Fuzzy Logic, Internet of Things.

INTRODUCTION

Edge computing that involves processing and analysis at the network edge close to data sources was enabled by the accelerated expansion of the Internet of Things. Statistical features or shallow machine learning methods have been leveraged as the basis for classical anomaly detection approaches.

Yet, as the variety and complexity of data in edge computing networks grow, increasingly sophisticated and intelligent solutions are needed [8]. Recent research has shown the viability of deep learning-based anomaly detection, which can identify complex patterns and relationships in the data [4–7]. This paradigm shift has introduced both new promise and challenges to anomaly detection, especially in deep learning. [8]. Each irregular pattern of activity or behavior that differs considerably

from the baseline set of normal network traffic and behavior is known as an anomaly in a computer network. Cyberattacks (e.g., intrusions, DDoS attacks, malware), system crashes (e.g., hardware errors, software bugs), configuration mistakes, sudden traffic spikes, or odd data transfer behavior are only a few of the issues that anomalies can indicate. Since it identifies malicious patterns or network traffic behavior patterns that may indicate the presence of potential security threats, performance anomalies, or operation faults, network anomaly detection is thus an indispensable part of network security and administration [7].

Point anomalies, contextual anomalies, and collective anomalies are the three broad categories under which anomalies in edge computing networks can be classified [6]. Individual data points that are a long way from the usual behavior of the system are referred to as point anomalies. Data points that are unusual in one context or environment but regular in another are referred to as contextual anomalies. The phenomenon of contextual anomalies, where a chunk of information seems unusual in one context but not necessarily in another, has been caused by the huge growth of the Internet of Things. A group of related data points that show abnormal behavior as a group but not as individual data points are called collective anomalies [10]. The next subsection simply defines the ideas of edge computing and deep learning.

1.1 Edge Computing

Edge computing is a new paradigm in the field of computing. It makes cloud computing service and utilities more accessible to the end user and is marked by swift processing and rapid application response time. Fast processing and rapid response time are necessary for applications currently developed with internet connectivity like surveillance, virtual reality, and real-time traffic monitoring [11-12]. End users typically execute these apps on their low-end mobile devices while the processing and main services are done on cloud servers. Using cloud services by mobile devices lead to mobility issues and high latency [13][14]. Edge computing addresses the aforementioned application needs by relocating the processing to the edge of the network.

Edge computing is not the same as conventional cloud computing. It is a new paradigm of computing that conducts computing on the edge of the network. Its fundamental principle is to bring computing nearer to the data source. Scholars have various definitions of edge computing. In distributed edge computing, a model for anomaly detection is initially developed using a training set because of the limitations of traditional anomaly detection methods, including their inability to scale and adapt. Subsequently, a dimension correlation-based anomaly identification method is proposed that is able to detect anomalies in both single-source and multi-source time series. Most specifically, the reliability of the results of detection is provided by monthly refreshes to the anomaly rules base [15].

1.1.1 Edge Computing Network Anomalies:

Edge computing is susceptible to network anomalies due to the special nature and issues of distributed computing at the network edge. The primary contribution is the proposal of an edge computing architecture that can detect anomalies in numerous sources across multiple endpoints [31].

- Edge Network Congestion: There are a number of reasons why edge networks at the edge can get congested. Greater data traffic, bandwidth limitations, network infrastructure bottlenecks, poor routing, and bursty traffic congestion are a few of the primary causes of edge network congestion, which can cause packet loss, increased latency, and degraded performance. Issues with edge network congestion can be handled in various ways [].
- Latency Spikes: In edge computing, sudden and substantial spikes in how long it takes for data to travel between processing or storage assets and edge devices are called latency spikes. The responsiveness and performance of applications at the network edge can be negatively impacted by these spikes, which could arise from a host of different origins. The processing of information at the edge of the network, near where the data was created, is called edge computing. However, due to lack of computer resources or network congestion at the edge network, latency bursts may lead to delays in reaction times as well as processing of data [].

- Edge Infrastructure Failures: Inadequate security frameworks, software and firmware vulnerabilities, insider threats, and connectivity threats Distributed architecture, limited resources, some common factors are physical vulnerability, connectivity danger, and insider threats. Edge devices and networks are vulnerable to security threats. Cloud computing can have high security and privacy protection measures in place, but edge computer nodes generally process at slower speeds [32].
- Resource Constraints: Several features of the edge environment, including power constraints, limited processing power, storage space, memory limitations, network bandwidth, diverse hardware, and scalability issues, can result in resource constraint issues in edge computing. Edge devices tend to have restricted computation, memory, and energy capabilities. The overall performance of the edge network can be affected by anomalies caused by the depletion or lack of proper maintenance of these resources.
- Communication Failures: Edge devices and nodes can also sometimes have connectivity problems or communication failure due to volatile network connections or interference. Such errors might delay data transmission and lead to service outages. Network instability is another cause of communication failure. In order to reduce such communication failures, constructing the edge network with redundancy, giving higher priority to critical data, maximizing network capacity, implementing strong error handling and recovery mechanisms, and regularly monitoring and revising the network architecture is very important. A failure of one node could lead to a series of cascade failures that compromise service delivery and hinder the achievement of certain goals.

The common types of anomalies in Edge computing are shown in Fig. 1.

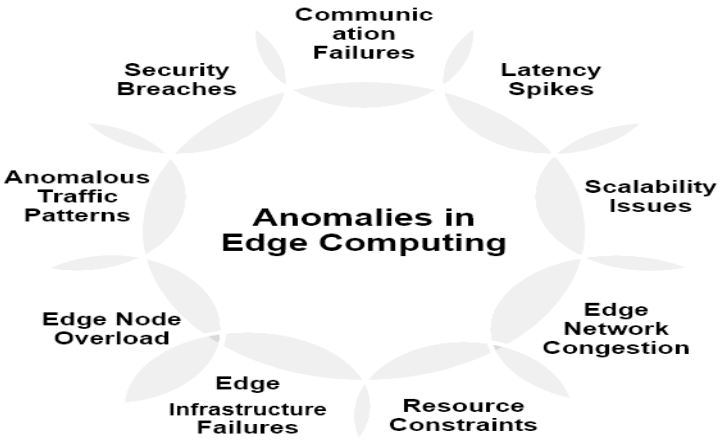

Fig. 1. Common anomalies in edge computing network

1.2 Deep Learning

Deep learning is an AI branch of machine learning that has risen as a revolution technology. Modeling human thought processes, it lies on artificial neural networks (ANNs), allowing machines to make inferences out of enormous quantities of data. Several fields, including image recognition, natural language processing, medicine, finance, and autonomous systems, have seen phenomenal success with deep learning models, especially deep neural networks (DNNs) [27]. Generalization over a wide range of problem domains with minimal feature engineering is one of the defining features of deep learning. Despite this, issues like interpretability, computational cost, and efficiency of data are research agendas [28].

A powerful tool for identifying anomalies across many areas of application, including edge computing, is deep learning. Deep learning methods fall into one of four categories: supervised, unsupervised, hybrid, and one-class neural networks.

Some well-known algorithms for deep learning-based anomaly detection are listed below:

1.2.1 Auto encoders: Artificial neural networks known as autoencoders are often used in unsupervised learning tasks, particularly anomaly detection. Their task is to learn a compressed representation (encoding) of the input data and then reconstruct it accurately. The basic concept behind their application in anomaly detection is that they find it difficult to reconstruct unusual or novel data after learning the patterns of normal data during training, resulting in higher reconstruction errors [29]. The two main components of an autoencoder are a decoder that decodes the input from this representation and an encoder compressing input data into a low-dimensional latent space. A signal of anomalies is the reconstruction error, which is often measured by metrics such as Mean Squared Error (MSE) or Mean Absolute Error (MAE). Since they deviate from the normal patterns learned during training, inputs with high reconstruction errors are labeled as potential anomalies [30].

1.2.2 Convolutional Neural network (CNNs): In various computer vision applications, including object recognition, image segmentation, and image classification, Convolutional Neural Networks (CNNs) have shown to excel. They are particularly ideal for these purposes due to their ability to learn spatial hierarchies of features automatically and adaptively from the incoming data. using network traffic patterns, CNNs were successfully deployed to detect anomalies for network security purposes. Through the time-series nature of the network traffic, CNNs could learn to discern trends and recognize anomalies or potential threats. As an instance, it has been proposed that a CNN-enabled real-time anomaly detection approach extracts statistical details of network traffic to detect and study anomalous traffic efficiently [31].

1.2.3 Recurrent Neural network (RNNs): An artificial neural network class known as Recurrent Neural Networks (RNNs) was developed specifically to deal with sequential input. In contrast to standard feedforward neural networks, RNNs have a mechanism that enables them to keep track of previous inputs, which makes them suitable for applications that require time-series analysis, speech recognition, natural language processing (NLP), and anomaly detection.RNNs can analyze network traffic patterns over time and detect anomalies that can indicate malicious activity. RNNs learn sequential financial transactions to detect suspicious activity. RNNs analyze sensor data in predictive maintenance to detect problems before system failures occur [32].

1.2.4 Generative Adversarial Networks (GANs): GANs are significant in numerous fields and have grown to be a powerful tool in deep learning for generating realistic data. To make them more effective, continued research seeks to reduce mode collapse, improve stability, and develop evaluation metrics. Normal network edge behavior can be found through the distribution of GANs. The generator network is trained to produce fabricated normal data, but the discriminator network distinguishes between forged and genuine data. Anomalies may be detected by measuring the discriminator's power to categorize new data samples. In the field of anomaly detection, researchers are dedicated to correctly and successfully detecting abnormal images in real-world applications. [33].

1.2.5 Variational Autoencoders (VAEs): Data generation is modeled probabilistically with VAEs, a type of generative autoencoder. VAEs learn the parameters of a probability distribution (e.g., Gaussian) in the latent space instead of a direct encoding. Samples in the latent space that are far from the learned distribution are identified as anomalies.In edge computing, the VAEs are utilized to track the distribution of network activity and resource usage. Variational autoencoders (VAEs), which are a type of generative deep learning model, are able to learn from multiple forms of data, including text and images. In detection of unknown threats, VAEs that replace the neural network posteriors tend to perform better compared to autoencoders and one-class support vector machines.

1.2.6 Long Short-Term Memory Networks

One type of recurrent neural network used to process sequence input is the Long Short-Term Memory network (LSTM). LSTMs are useful in time series because they contain memory cells that allow them to store information for hundreds or thousands of time steps. Language modeling, machine translation, and a whole host of related tasks are some of the numerous long sequence

learning tasks that LSTMs can be applied to. As the output travels through feedback loops for one input, the LSTM network architecture may prevent it from deteriorating (long-term dependence) or exploding. Beyond earlier deep convolutional neural networks, the network can identify sequential patterns due to the feedback loops [2]. DFHNNLM is introduced on the basis of inter-class and intra-class fuzzy membership measures with maximum data point's coverage to construct the FSHs in the hidden layer of FHNN. The hidden to output layer weights of FHNN are learned in parallel while constructing FSHs in the hidden layer [1] [3].

This study employs a DFHNNLM in proposing a new framework for edge computing network anomaly identification. a deep learning architecture that unites the strength of neural networks and fuzzy logic. The proposed approach attempts to address the challenges of edge computing anomaly detection from real-time sensor data.

1 BACKGROUND AND RELATED WORK

The most contemporary machine learning and deep learning techniques for anomaly detection in computer networks and the Internet of Things are addressed in this chapter. In recent years, there has been a tremendous amount of work concentrated on anomaly detection on edge computing networks. Various techniques, such as machine learning, deep learning, and statistical approaches, have been explored in prior studies. It has also been proven that anomaly detection using time-series data and transport networks is most effective when generative adversarial networks and autoencoders are implemented [7] [17]. Further, the research proposes a deep learning-based end-to-end anomaly detection system for transportation networks that combines traditional anomaly detection techniques with deep neural networks. Following are the various literature that points out the concepts in this research work.

Zakariah, M., AlQahtani, S.A., Alawwad etal. has proposed an Intrusion Detection System (IDS) using Artificial Neural Networks (ANNs) for enhancing network intrusion detection [18]. Key points include robust data preprocessing, PCA-based feature selection, and optimal hyper parameter tuning. The research points to the capability of ANNs in identifying complex patterns of network traffic and anomalies. It acknowledges the limitations of the dataset, but stresses further research in dynamic network settings for improved security. Xu, W., Jang-Jaccard et al., has proposed an advanced model of network anomaly detection based on autoencoders [19]. The proposed model is to be used for overcoming vulnerabilities in network security. The proposed 5-layer autoencoder performs extremely well on the NSL-KDD dataset, where accuracy and F1 values stand at 90.61% and 92.26%, respectively. The authors emphasize the importance of using an effective reconstruction error function and a new data preprocessing mechanism for enhanced feature learning and dimensionality reduction. The study highlights the promise of the model in augmenting network intrusion detection accuracy, also including the challenges posed by the limited publicly available intrusion data. Kasongo, S. M. has proposed NSL-KDD results showed, the XGBoost-LSTM model outperformed other traditional machine learning algorithms with a test accuracy (TAC) of 88.13%. The LSTM results showed a remarkable TAC of 85.93%, while the GRU yielded 85.65% [20]. The Simple RNN was always quicker to train than the other models and would thus be an excellent choice for resource-constrained environments. Performance measures are detailed in different tables, underlining how these models perform in binary and multiclass classification problems. C. Ieracitano, A. Adeel, F. C. Morabito et al. has analyzed & implemented the NSL-KDD dataset utilizing a simple autoencoder architecture with three layers. The classification accuracy produced in binary was 84.21%, which indicates the potential of autoencoders for anomaly identification [21]. Su, T., Sun, H., Zhu, J., Wang, S. et al. has proposed BAT-MC is a deep neural network model which combines BLSTM and an attention mechanism for the task of intrusion detection. It achieves 84.25% accuracy on the NSL-KDD dataset, improving feature extraction and reducing reliance on manual feature engineering for detecting network anomalies [22].

2 METHODOLOGY

The Proposed Anomaly detection Model for Edge computing is explained in section 3.1 followed by description of experimental NSL-KDD dataset in section 3.2. The Architecture of DFHNNLM is explained in section 3.3. Section 3.4 covers the detailed working of proposed DFHNNLM algorithm followed by an evaluation metrics in section 3.5.

2.1 Proposed Anomaly detection Model for Edge Computing

The proposed model for anomaly detection composed of multilayered architecture which is shown in Fig. 2. The diagram illustrates a multi-layered system for anomaly detection on IoT data processed by edge computing using DFHNNLM algorithm. It starts with an IoT Sensor Layer where devices collect data, which is then aggregated and stored in the cloud via an Edge Computing Platform. The Deep Learning Layer processes this data, beginning with importing the dataset, preprocessing it, extracting relevant features, building a model using a DFHNNLM, and finally performing anomaly detection. The Anomaly Detection stage categorizes anomalies into point, contextual, or collective types. Normal outcomes lead to test result acceptance, while anomalous outcomes trigger the implementation of measures to address the anomalies.

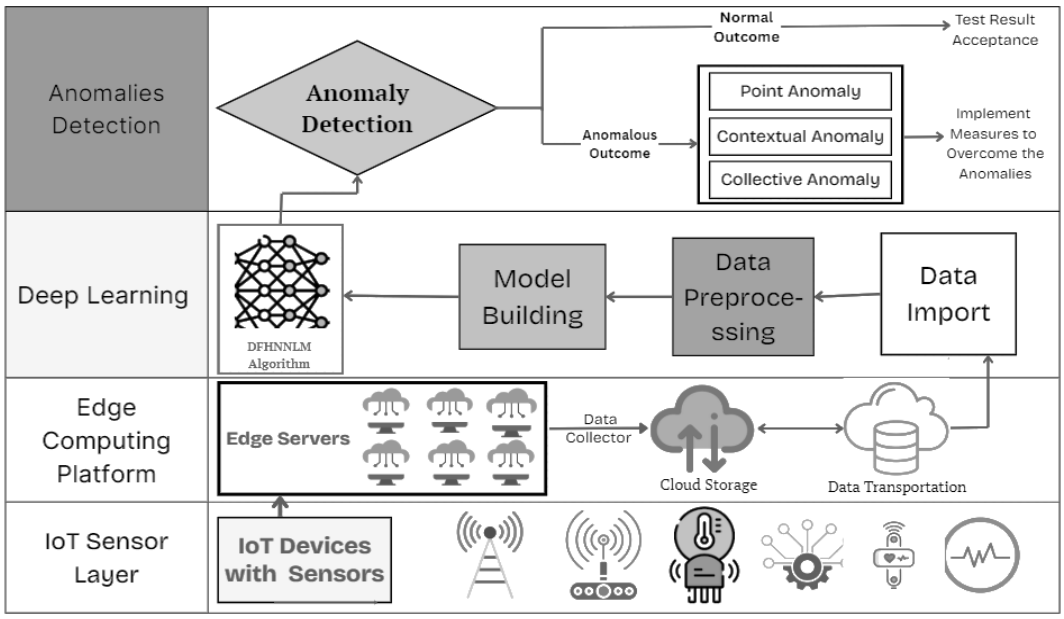

Fig. 2. Proposed model for Anomaly detection

The block diagram has four main layers. The layer by layer description of diagram is as follows

a) IoT Sensor Layer:

This layer represents the origin of the data. It consists of various Internet of Things (IoT) devices, each equipped with sensors. These sensors collect data related to their specific environment or function. The different sensors include Wireless Communication Antennas that collect the data related to network connectivity or signal strength, Smart Meters used to measure energy consumption, water usage, etc., Temperature Sensors to monitor temperature in a specific area, Heart Rate Monitors used for collecting health-related data and so on.

b) Edge Computing Platform

This layer showcases the use of edge computing over the Edge Servers. Edge servers are located closer to the IoT devices, enabling faster data processing and reduced latency. The diagram illustrates multiple edge servers working in parallel. The Data Collector gathers the data streamed from the IoT devices. The cloud Storage uses the collected data which is then transmitted to the cloud for persistent storage and further analysis. The Dataset Formation stage indicates the organization and structuring of the raw data into a usable dataset for the deep learning model.

c) Deep Learning Layer

This layer involves importing the dataset by loading the prepared dataset from the cloud storage. The Data Preprocessing involves cleaning and transforming the raw data to make it suitable for the DFHNNLM algorithm. The relevant features are extracted from the preprocessed data. These features are the most informative aspects of the data that the model will use to learn patterns. Finally, the model Building stage involves applying the DFHNNLM algorithm to check anomaly detection.

d) Anomaly Detection

This is the core layer of the system in which the model analyzes the data and classifies it as either "Normal Outcome" or "Anomalous Outcome". The normal outcome involves the data is deemed normal and the "Test Result Acceptance" is triggered, indicating that the system is functioning as expected. In case of Anomalous Outcome, it's further categorized into Point Anomaly, Contextual Anomaly or Collective Anomaly. Therefore, if the anomaly is detected then there's need to implement the measures to overcome the Anomalies.

2.2 NSL-KDD Dataset

The NSL-KDD dataset has 125,973 records in the KDDTrain+ subset (67,343 normal and 58,630 abnormal examples) and 22,544 records in the KDDTest+ subset (9,711 normal and 12,833 abnormal examples). The data set consists of records, with each record having 41 features that are categorized as numeric and categorical types. 38 numeric features observe different attributes of network traffic such as duration, source and destination bytes, connection count, and so on. In addition, there are three categorical features: Protocol Type (TCP, UDP, ICMP), Service (which includes 70 unique attributes), and Flag (which includes 11 unique attributes). For ease of analysis and model training, these categorical features are usually transformed into numerical values in pre-processing. Once all the features are merged, a total of 122 features are generated.

2.3 Architecture of DFHNNLM

A DFHNNLM is a type of neural network model that incorporates fuzzy logic to control classification uncertainty and employs hyperspheres to depict clusters in high-dimensional space. When data points are not clearly separable, it is particularly advantageous [11] [12]. The architecture of DFHNNLM is shown in Fig. 2.

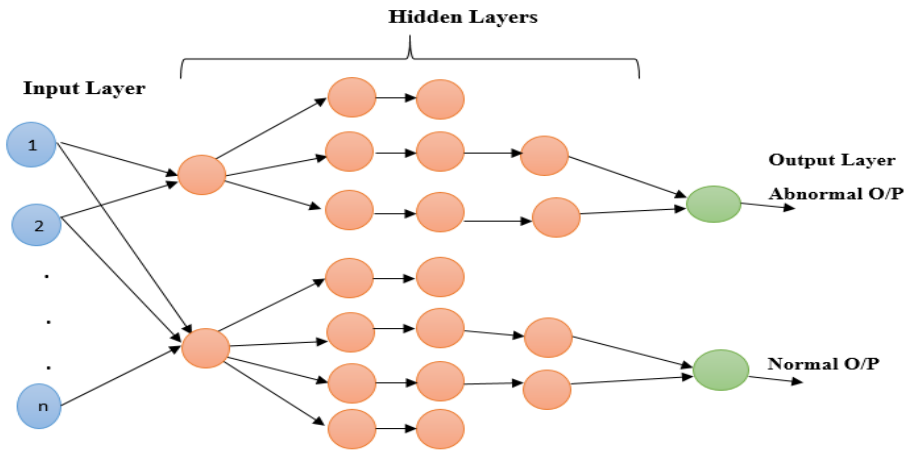

Fig. 3. Architecture of DFHNNLM

The architecture consists of six layers which are broadly described as follows.

a) **Input Layer (First Layer)**- The input layer consists of input neurons label as 1 to n. This neuron receives anomalous or non-anomalous input data from sensors and passes it to the next layer.

b) **Hidden Layer 1 (Second Layer)** - The second layer is first hidden layer which is created to form the two clusters normal (Non-anomalous) & abnormal (anomalous). We perform the recall test and accordingly move to the next hidden layer because first hidden layer doesn't give the efficiency of 100%.

c) **Hidden layer 2 (Third Layer)**- In this layer, we create the fuzzy hyperspheres (FHS) with the proposed algorithm in which the radius is equal to the actual intraclass distance. This layer provides the initial number of FHS that are created for DFHNN.

d) Hidden Layer 3 (Fourth Layer)- In this layer, we create the fuzzy hyperspheres (FHS) with the proposed algorithm in which the radius is equal to the actual distance of the centroid with clustered class patterns. This layer provides the initial number of FHS with precision radius that are created for DFHNN

e) **Hidden Layer 4 (Fourth Layer)**- This layer is the extension of hidden layer 3, where the FHS having no patterns are discarded. This layer will help in improving the generalization efficiency by reducing the pattern space.

f) **Output layer (Final Layer)**- The output layer is nothing but class layer which consists of two neurons, one for anomalous and other for Normal output [12].

All the hidden layers consist of neurons characterizes by fuzzy membership function

$$mem_j(length, radius_j) = \begin{cases} 1 & length \leq radius_j \\ radius_j/length & Otherwise \end{cases}$$

The second, third and fourth and fifth layers, which are the hidden layers of DFHNNLM architecture, are created by the algorithm stated in section 3.3.

2.4 Proposed DFHNNLM Algorithm

The algorithm consists of two stages. The first stage creates two FHS based on the intraclass and fuzzy membership function to cluster its own class patterns which represent the second layer of DFHNNLM.

Algorithm for Stage 1:

Step 1. Find the pattern of the precise class which will cluster all the patterns of own class with appropriate radius by calculating the intraclass distance and fuzzy membership function with considering the overlap to other classes.

Step 2. Repeat the same for all classes

Step 3. Determine the accuracy for the patterns in the dataset

Step 4. If the accuracy is not desirable, then go to stage 2

Since the efficiency of stage 1 is not desirable, then the additional layers need to be introduced. The algorithm for stage 2 i.e. creating second, third and fourth hidden layer is explained below. While creating the first hidden layer in stage 1, the links with output layer are also updated. The same is adopted in stage 2 which is given below.

Algorithm for Stage 2:

Step 1: Compute the Intraclass-Class Distance Matrix

Step 2: Create a second hidden layer by making radius is equal to the actual intraclass distance & group the patterns of same class using Fuzzy membership function. Repeat the process for the class till all the patterns are clustered. Continue the process for all the classes.

Step 3: Create a third hidden layer by making the radius is equal to the actual distance of the centroid with clustered class patterns. This layer provides the initial number of FHS with precision radius that are created for DFHNN

Step 4: Create a fourth hidden layer which is an extension of Hidden Layer 3, the Fuzzy Hyperspheres (FHS) with no associated patterns are pruned to reduce the pattern space, thereby enhancing generalization efficiency.

Step 5: Always class output node in output layer is created at the start of training

Step 6: Update the weights between fourth hidden Layer and output layer

All the FHS are governed by the membership function as stated below

$$mem_j(length, radius_j) = \begin{cases} 1 & length \leq radius_j \\ radius_j/length & Otherewise \end{cases}$$

And the weight connections between fourth layer and output layer are binary given by the formula

$$O_{jk} = \begin{cases} 1 & if\ H_j\ is\ a\ Hypersphere\ of\ class\ c_k \\ 0 & Otherewise \end{cases}$$

While testing, the output is determined by using

$$C_k = \max_{j=1}^{J} mem_j\ O_{jk} \qquad k = 1,2,\ldots.,K$$

Where K is the number of classes.

2.5 Evaluation Metrics

The performance of deep learning models is measured using four parameters namely Accuracy, Precision, Recall and F1 Score. A confusion matrix visualizes and summarizes the performance of a classification algorithm. It illustrates the overall picture of classification performance that consists of True Positive (TP), False Positive (FP), True Negative (TN), and False negative (FN) [40][45]. The representation of Confusion matrix is shown in Fig 4.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Fig. 4. Representation of Confusion matrix

The performance metrics are defined as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1 - Score = 2 \text{ x } \frac{Precision \text{ x } Recall}{Precision + Recall} \quad (4)$$

3 RESULTS AND DISCUSSION

The existing results proposed by researchers are compared with proposed DFHNN learning model. The comparison of results between proposed model and existing models are given in Table 1 and the graphically represented in Fig. 5.1.1 for Accuracy, Fig. 5.1.2 for Precision, Fig. 5.1.3 for Recall and Fig. 5.1.4 for F1-score.

Table 1. Comparison of results between proposed model and existing models

Model	Accuracy	Precision	Recall	F1-Score
Proposed DFHNNLM	**0.977**	**0.970**	**0.976**	**0.973**
Sparse Autoencoder+SVM [18]	0.84	0.96	0.76	0.85
Autoencoder [21]	0.84	0.87	0.80	0.81
DNN [35]	0.94	0.91	0.92	0.77
CNN [36]	0.95	0.94	0.99	0.97
CNN+LSTM [36]	0.96	0.94	0.99	0.97
LSTM [37]	0.89	0.97	0.97	0.95

From Table 1 and Fig. 5.1.1 to Fig.5.1.4, the comparative analysis of anomaly detection models reveals that the proposed DFHNNLM achieves the highest overall performance, with 97.7% accuracy, 97.0% precision, 97.6% recall, and a 97.3% F1-score. While CNN and CNN+LSTM [36] also show strong results, both attaining an F1-score of 97% and their accuracy remains slightly lower at 95% and 96%, respectively.

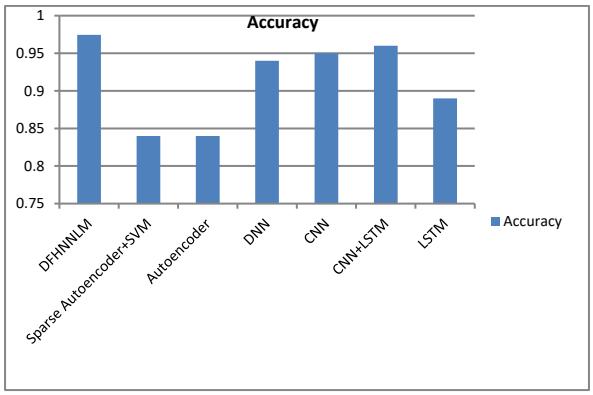

Fig. 5.1.1. Accuracy

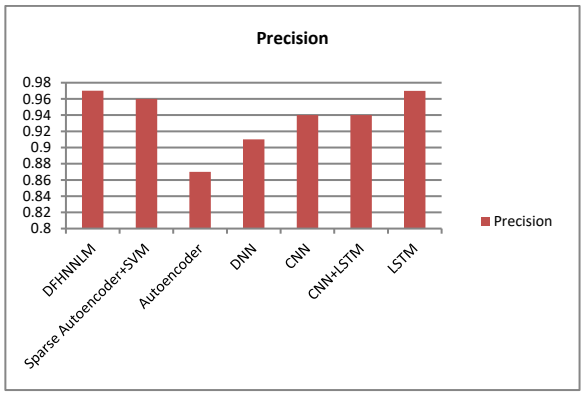

Fig. 5.1.2 Precision

The LSTM model [37] matches the highest precision and recall (97%) but trails in accuracy (89%) and F1-score (95%). In contrast, traditional models like Sparse Autoencoder + SVM [18] and Autoencoder [21] display moderate performance with 84% accuracy and lower recall and F1-scores. The DNN [35] shows better accuracy at 94% but underperforms in F1-score (77%). These findings clearly demonstrate the DFHNNLM's superior ability to deliver balanced and accurate anomaly detection in edge computing scenarios.

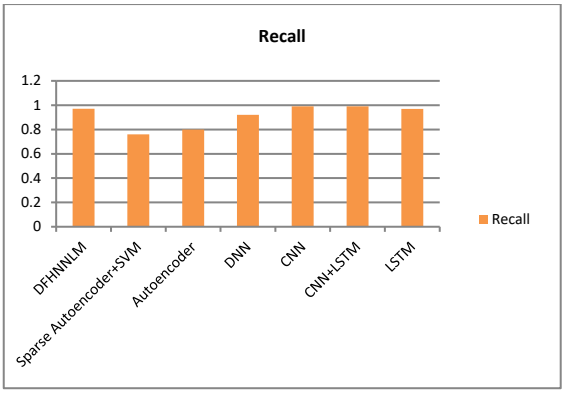

Fig. 5.1.3. Recall

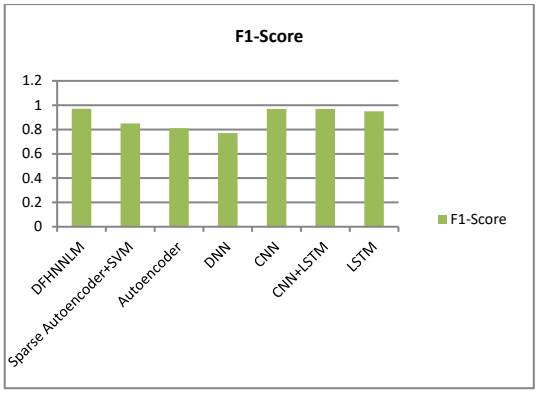

Fig. 5.1.4. F1-Score

The proposed DFHNNLM for anomaly detection in edge computing networks shows promising results in the experimental evaluation. The key findings are:

1. The experimentations reveal that DFHNNLM classifier is a fast converging network having 100% efficiency for the training set and appreciable test efficiency in comparison with other classifiers [1].
2. The DFHNNLM achieves higher accuracy in anomaly detection compared to traditional methods, with a true positive rate of over 97% and a low false positive rate.
3. The DFHNNLM demonstrates robustness to noise and uncertainty in the edge computing data, due to the integration of fuzzy logic pre-processing.
4. The adaptive nature of the Deep Fuzzy Hyper Neural Network learning model, facilitated by the fuzzy logic components, enables the framework to adapt to changing network conditions and maintain high performance over time.

The results validate the effectiveness of the proposed approach and its potential for real-world deployment in edge computing networks.

The experimental results demonstrate the superiority of the Deep Fuzzy Hyper Neural Network over traditional anomaly detection methods, highlighting its potential for real-world deployment in edge computing applications. Future research directions may include exploring the integration of the Deep Fuzzy Hyper Neural Network with other edge computing technologies, such as distributed computing and edge analytics, to further enhance the performance and applicability of the framework.

4 CONCLUSION

In this research **work**, a novel Deep Fuzzy Hypersphere Neural Network Learning Model (DFHNNLM) is proposed for effective anomaly detection within edge computing networks. The model leverages the concept of fuzzy hyperspheres, constructed based on both inter-class and intra-class fuzzy membership metrics, to ensure maximal coverage of the input feature space. This approach enhances the model's ability to distinguish between normal and anomalous data distributions. The learning process is divided into two distinct stages

In the second stage, cluster formation is guided by the spatial positioning of patterns from different classes, rather than by the geometric width of the clusters. Since the positional attributes of patterns remain constant, the model becomes insensitive to the sequence in which training data is presented, thus eliminating order-dependency in learning. A key contribution of the proposed DFHNNLM is the synergistic integration of fuzzy logic principles with the representational depth of deep neural networks. This hybridization allows the model to robustly handle data that is inherently noisy, uncertain, and dynamically evolving challenges that are particularly prevalent in edge computing environments. Unlike traditional classifiers that rely on shortest-distance measures for pattern inclusion, the DFHNNLM utilizes fuzzy membership functions to evaluate the significance of each pattern. As a result, classification decisions are influenced more by the fuzzy hypersphere's width

than by proximity metrics. This paradigm shift leads to a model that demonstrates high classification precision, rapid convergence, and excellent training performance, while also achieving substantial generalization capability during testing.

REFERENCES

[1] Kulkarni, A. B., Bonde, S. V., & Kulkarni, U. V.: Class-Specific Fuzzy Hypersphere Neural Network. Procedia Computer Science 143, 285–294 (2018).

[2] Roy, M., Majumder, S., Halder, A., & Biswas, U.: ECG-NET: A deep LSTM autoencoder for detecting anomalous ECG. Engineering Applications of Artificial Intelligence 124, (2023).

[3] Kulkarni, A., & Kulkarni, N.: Fuzzy Neural Network for Pattern Classification. Procedia Computer Science 167, 2606–2616 (2020).

[4] Alamr, A., & Artoli, A. (2023). Unsupervised Transformer-Based Anomaly Detection in ECG Signals. In Algorithms (Vol. 16, Issue 3, p. 152). Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/a16030152

[5] Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.1901.03407

[6] Choi, J., Park, J., Japesh, A., & Adarsh, A. (2023). A Subspace Projection Approach to Autoencoder-based Anomaly Detection. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2302.07643

[7] Sharmila, V., Kannadhasan, S., Kannan, A. R., Sivakumar, P., & Vennila, V. (Eds.). (2024). *Challenges in Information, Communication and Computing Technology: Proceedings of the 2nd International Conference on Challenges in Information, Communication, and Computing Technology (ICCICCT 2024), April 26th & 27th, 2024, Namakkal, Tamil Nadu, India*. CRC Press. https://doi.org/10.1201/9781003559085

[8] Kabaivanov, S., & Markovska, V. (2020). Hybrid deep-learning analysis for cyber anomaly detection. In IOP Conference Series Materials Science and Engineering (Vol. 878, Issue 1, p. 12029). IOP Publishing. https://doi.org/10.1088/1757-899x/878/1/012029

[9] K. Cao, Y. Liu, G. Meng and Q. Sun, "An Overview on Edge Computing Research," in IEEE Access, vol. 8, pp. 85714-85728, 2020, doi: 10.1109/ACCESS.2020.2991734.

[10] Jayasinghe, M., Seneviratne, S., Seneviratne, A.: Deep learning for anomaly detection in network traffic data. In: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops).

[11] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, M. Imran, The role of edge computing in internet of things, IEEE Communications Magazine (99) (2018) 1–6.

[12] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, M. Song, Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Transactions on Wireless Communications 18 (1) (2019) 695–708.

[13] E. Ahmed, A. Akhunzada, M. Whaiduzzaman, A. Gani, S. H. Ab Hamid, R. Buyya, Networkcentric performance analysis of runtime application migration in mobile cloud computing, Simulation Modelling Practice and Theory 50 (2015) 42–56.

[14] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, A. Liotta, An edge-based architecture to support efficient applications for healthcare industry 4.0, IEEE Transactions on Industrial Informatics 15 (1) (2019) 481–489.

[15] X. Yu, X. Yang, Q. Tan, C. Shan, and Z. Lv, "An edge computing based anomaly detection method in IoT industrial sustainability," *Appl. Soft Comput.*, vol. 129, p. 109486, 2022. doi: 10.1016/j.asoc.2022.109486.

[16] Wu, X.; Wu, J.; Cheng, B.; Chen, J. Neural Network Based Situation Detection and Service Provision in the Environment of IoT. In Proceedings of the 2013 IEEE 78th Vehicular Technology Conference (VTC Fall),Wynn, LV, USA, 2–5 September 2013; pp. 1–5.

[17] Lee, S., Jin, H., Nengroo, S. H., Doh, Y., Lee, C., Heo, T., & Har, D. (2021). Smart Metering System Capable of Anomaly Detection by Bi-directional LSTM Autoencoder. In arXiv, Cornell University. https://doi.org/10.48550/arxiv.2112.03275.

[18] Zakariah, M., AlQahtani, S.A., Alawwad, A.M., & Alotaibi, A.A. (2023). *Intrusion Detection System with Customized Machine Learning Techniques for NSL-KDD Dataset*. Computers, Materials & Continua, 77(3), 4025–4054. Springer. https://doi.org/10.32604/cmc.2023.043752

[19] Xu, W., Jang-Jaccard, J., Singh, A., & Yuanyuan. (2021). Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. *IEEE Access*, 9, 134563–134576. https://doi.org/10.1109/ACCESS.2021.3116612

[20] Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. *Computer Communications, 199*, 113-125. https://doi.org/10.1016/j.comcom.2022.12.010

[21] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, ``Anovel statistical analysis and autoencoder driven intelligent intrusion detection approach," Neurocomputing, vol. 387, pp. 51_62, Apr. 2020.

[22] Su, T., Sun, H., Zhu, J., Wang, S., & Li, Y. (2020). BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *Springer Journal Name*, *Volume*(Issue), page numbers. https://doi.org/10.1109/ACCESS.2020.2972627

[23] Yuhuai Peng. Aiping Tan. Jingjing Wu. Yuanguo BI.: Hierarchical Edge Computing: A Novel Multi-Source Multi-Dimensional Data Anomaly Detection Scheme for Industrial Internet of Things. Open Access Journal IEEE.VOLUME 7.2019 August 23..pp. 111257- 111270.

[24] Anantha, A.P., Daely, P.T., Lee, J.M., Kim, D.S., 2020. Edge Computing-Based Anomaly Detection for Multi-Source Monitoring in Industrial Wireless Sensor Networks. In: ICTC 2020, (Springer)pp. 1890-1892.

[25] Hu, P., & Chen, W. (2019). *Software-Defined Edge Computing (SDEC): Principles, Open System Architecture and Challenges*. In IEEE SmartWorld, Ubiquitous Intelligence & Computing (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). Springer.

[26] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

[27] Samek, W., Montavon, G., & Müller, K. R. (2021). Towards interpretable machine learning: Theory and practice. *Proceedings of the IEEE, 109*(5), 612–634. https://doi.org/10.1109/JPROC.2021.3060483

[28] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504–507. https://doi.org/10.1126/science.1127647

[29] Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. *Proceedings of the MLSDA 2014: 2nd Workshop on Machine Learning for Sensory Data Analysis*, 4. https://doi.org/10.1145/2689746.2689747

[30] Liu, H., & Wang, H. (2023). Real-time anomaly detection of network traffic based on CNN. *Symmetry, 15*(6), 1205. https://doi.org/10.3390/sym15061205

[31] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[32] Zhang, L., Fan, F., Dai, Y., He, C.: Analysis and research of generative adversarial network in anomaly detection. In: 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP), (2022).

[33] M. Al-Qatf, Y. Lasheng, M. Al-Habib and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection", *IEEE Access*, vol. 6, pp. 52843-52856, 2018.

[34] C. Ieracitano, A. Adeel, F. C. Morabito and A. Hussain, "A novel statisticalanalysis and autoencoder driven intelligent intrusion detectio approach", *Neurocomputing*, vol. 387, pp. 51-62, Apr. 2020.

[35] Gbashi, Ekhlas K."Intrusion Detection System for NSL-KDD Dataset Based on Deep Learning and Recursive Feature Elimination." Engineering and Technology Journal, vol. 39, no. 7, 2021, article 1695. https://doi.org/10.30684/etj.v39i7.1695.

[36] Harini, R., Maheswari, N., Ganapathy, S., & Sivagami, M. (2023). An effective technique for detecting minority attacks in NIDS using deep learning and sampling approach. Ain Shams Engineering Journal. https://doi.org/10.1016/j.aej.2023.07.063.

[37] S. Naseer *et al*., "Enhanced Network Anomaly Detection Based on Deep Neural Networks," in *IEEE Access*, vol. 6, pp. 48231-48246, 2018, doi: 10.1109/ACCESS.2018.2863036.