Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.046253

ARTICLE

Traffic-Aware Fuzzy Classification Model to Perform IoT Data Traffic Sourcing with the Edge Computing

Huixiang Xu*

College of Information Engineering, Zhengzhou University of Technology, Zhengzhou, 450044, China

*Corresponding Author: Huixiang Xu. Email: zzdxxhx2@163.com

Received: 25 September 2023 Accepted: 19 December 2023 Published: 27 February 2024

ABSTRACT

The Internet of Things (IoT) has revolutionized how we interact with and gather data from our surrounding environment. IoT devices with various sensors and actuators generate vast amounts of data that can be harnessed to derive valuable insights. The rapid proliferation of Internet of Things (IoT) devices has ushered in an era of unprecedented data generation and connectivity. These IoT devices, equipped with many sensors and actuators, continuously produce vast volumes of data. However, the conventional approach of transmitting all this data to centralized cloud infrastructures for processing and analysis poses significant challenges. However, transmitting all this data to a centralized cloud infrastructure for processing and analysis can be inefficient and impractical due to bandwidth limitations, network latency, and scalability issues. This paper proposed a Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) for traffic data analysis. The proposed techniques effectively utilize clustering and classification procedures to improve classification accuracy in analyzing network traffic data. SLItFC addresses the intricate task of efficiently managing and analyzing IoT data traffic at the edge. It employs a sophisticated combination of fuzzy clustering and self-learning techniques, allowing it to adapt and improve its classification accuracy over time. This adaptability is a crucial feature, given the dynamic nature of IoT environments where data patterns and traffic characteristics can evolve rapidly. With the implementation of the fuzzy classifier, the accuracy of the clustering process is improvised with the reduction of the computational time. SLItFC can reduce computational time while maintaining high classification accuracy. This efficiency is paramount in edge computing, where resource constraints demand streamlined data processing. Additionally, SLItFC's performance advantages make it a compelling choice for organizations seeking to harness the potential of IoT data for real-time insights and decision-making. With the Self-Learning process, the SLItFC model monitors the network traffic data acquired from the IoT Devices. The Sugeno fuzzy model is implemented within the edge computing environment for improved classification accuracy. Simulation analysis stated that the proposed SLItFC achieves 94.5% classification accuracy with reduced classification time.

KEYWORDS

Internet of Things (IoT); edge computing; traffic data; self-learning; fuzzy-learning

1 Introduction

In the ever-evolving landscape of technology, the Internet of Things (IoT) has emerged as a transformative force, connecting the physical world to the digital realm like never before [1]. At the heart of this revolutionary concept lies IoT data sourcing, a critical process underpinning the entire IoT ecosystem. IoT data sourcing refers to acquiring, collecting, and aggregating vast amounts of data generated by interconnected devices and sensors scattered across diverse environments [2]. These intelligent devices, embedded in everyday objects and industrial machinery, tirelessly gather information on various aspects of our lives, industries, and the environment. As a result, IoT data sourcing plays a pivotal role in enabling informed decision-making, facilitating automation, and uncovering valuable insights that drive innovation across industries, ranging from healthcare and agriculture to manufacturing and smart cities [3]. In this dynamic and data-driven era, the significance of IoT data sourcing cannot be overstated, as it paves the way for a more connected, efficient, and intelligent world.

IoT comprises billions of interconnected devices that generate an unfathomable amount of data. The concept of edge computing has emerged as a game-changer [4]. Edge computing in IoT represents a paradigm shift that brings computational power and intelligence closer to the data source rather than relying solely on centralized cloud infrastructure. By leveraging the capabilities of edge devices and gateways deployed at the network's edge, this innovative approach enables real-time data processing, analysis, and decision-making, significantly reducing latency, enhancing Security, and optimizing bandwidth usage [5]. This symbiotic relationship between IoT and edge computing unfolds a new horizon of possibilities, empowering industries to create more intelligent, responsive, and autonomous systems. From autonomous vehicles and smart factories to remote healthcare and intelligent cities, the fusion of IoT and edge computing is reshaping how we perceive and harness data, leading us toward a future defined by unprecedented efficiency, scalability, and potential [6].

In our fast-paced world, urban congestion and traffic bottlenecks have become ubiquitous challenges, demanding innovative solutions to optimize transportation efficiency and alleviate the woes of daily commuters [7]. This is where Traffic-Aware Routing with IoT traffic sourcing steps in as a transformative force. This cutting-edge approach revolutionizes navigating urban landscapes by seamlessly integrating the IoT with traffic-sourcing technologies [8]. Traffic-aware routing leverages real-time data from IoT devices, such as smart sensors, cameras, and GPS-equipped vehicles, to dynamically analyze traffic patterns, road conditions, and congestion levels [9]. This wealth of information allows the routing algorithms to intelligently recommend the most efficient and least congested routes for drivers, cyclists, and pedestrians [10]. As a result, this innovative synergy of IoT traffic sourcing and intelligent routing not only optimizes travel times and reduces carbon emissions but also lays the foundation for creating more innovative, safer, and more sustainable transportation networks that cater to the needs of modern society [11].

While Traffic-Aware Routing with IoT traffic sourcing holds tremendous promise in revolutionizing urban transportation, its implementation also comes with complex challenges and issues [12]. With IoT seamlessly connecting our cities and vehicles, critical aspects demand careful consideration. First and foremost is the issue of data privacy and Security [13]. With many IoT devices collecting and transmitting real-time traffic information, the potential for data breaches and unauthorized access to sensitive information becomes a pressing concern. Ensuring robust encryption protocols and stringent access controls becomes paramount to protect users' privacy and maintain the integrity of the system [14].

Additionally, the reliability of the data collected from IoT devices becomes crucial, as inaccurate or corrupted information could lead to misguided routing decisions, exacerbating congestion rather than mitigating it [15]. Moreover, as the scale of IoT traffic sourcing increases, the sheer volume of data generated can strain network bandwidth and computing resources, requiring efficient data management strategies to handle the influx of information effectively [16]. Furthermore, coordinating and integrating multiple IoT devices and platforms from various vendors presents interoperability challenges, necessitating standardized protocols and seamless communication interfaces. The existing limitations observed with potential limitations could be the model's dependence on the quality and quantity of available IoT data. If the IoT devices do not generate sufficient data or the data is noisy or incomplete, it can negatively impact the classifier's performance. IoT networks can vary significantly in size, and the model should be able to handle large-scale deployments efficiently. This leads to scalability issues in the network. Depending on the Complexity of the model, it may require significant computational resources for real-time processing. This could limit its feasibility in resource-constrained IoT edge devices. IoT environments are dynamic, and devices may change or add new devices over time. Transitioning from a simulation or controlled environment to a real-world deployment can uncover unforeseen challenges and limitations that were not apparent during the development and testing phases.

The paper makes several significant contributions to the field of IoT data analysis and edge computing:

- The paper proposes a novel classification model, SLItFC, which combines fuzzy logic with a self-learning mechanism. This approach enables the model to adapt and refine its classification rules over time, improving its accuracy in analyzing IoT network traffic data.
- With edge computing capabilities, the SLItFC model performs data analysis at the network's edge, closer to the IoT devices generating the data. This reduces latency and bandwidth usage by minimizing the need to transmit data to a central server for analysis, making the system more efficient and responsive.
- The SLItFC model achieves a high clustering accuracy of 94.5% in classifying IoT network traffic data into distinct clusters. Integrating fuzzy logic and the self-learning process allows the model to handle uncertainties and variations in traffic patterns, resulting in more accurate and meaningful classifications.
- The paper provides insights into how edge computing and fuzzy logic can be combined to enhance IoT data analysis. This integration opens new possibilities for designing intelligent and adaptable systems in edge computing architectures.

SLItFC finds valuable applications in various domains where IoT devices generate substantial data volumes, demanding efficient data processing and analysis. In a smart city, for instance, where IoT sensors and cameras continuously monitor traffic conditions and environmental factors, SLItFC can classify and analyze real-time data at the edge. This empowers the city to make immediate decisions, such as traffic signal adjustments or route optimizations, while alleviating network congestion and latency associated with transmitting all data to a central cloud. Similarly, in industrial IoT for manufacturing, precision farming, healthcare wearables, or retail settings, the model's edge-based fuzzy classification enables real-time insights, predictive maintenance, patient care enhancements, and personalized customer experiences, all while reducing data transmission overhead and maintaining high classification accuracy. Such applications underscore the model's potential in optimizing processes and decision-making across diverse IoT-driven ecosystems. The paper's contribution lies in introducing the SLItFC model, a self-learning fuzzy classifier designed for IoT data clustering in edge computing environments. The model's ability to learn and adapt and its promising classification accuracy make it a valuable addition to the growing body of research in IoT analytics and edge computing. The insights gained from this work can inspire further developments and innovations in intelligent IoT data analysis systems.

The remaining sections of the paper are organized as follows: Section 2 describes the related work of the proposed method. Section 3 defines the system model. Section 4 explains self-learning internet traffic clustering. Classification with self-learning clustering is defined in Section 5. Simulation results are explained in Section 6. Finally, the conclusion is defined in Section 7.

2 Literature Survey

Traffic-aware routing with IoT traffic sourcing holds immense potential in transforming transportation efficiency. It is not without its challenges and complexities. As cities become more interconnected and data-driven, several critical issues arise in implementing this innovative approach [17]. One of the primary concerns is data privacy and Security. Collecting and utilizing real-time data from IoT devices necessitates handling sensitive information, such as location data and travel patterns, raising apprehensions about potential breaches and unauthorized access. Furthermore, the reliability and accuracy of the IoT-generated data become paramount, as any inaccuracies or delays in data transmission can lead to misguided routing decisions and exacerbate traffic issues [18]. Additionally, ensuring seamless interoperability and standardization among diverse IoT devices and systems is another hurdle, as different manufacturers and technologies might follow distinct protocols and communication standards. Moreover, the rapid proliferation of IoT devices and the exponential growth of data volumes demand robust infrastructure and computational capabilities to effectively process and analyze data in real time. Finally, balancing personalized routing recommendations and overall traffic optimization for the greater good poses a significant ethical challenge.

Zhan et al. [19] explored multi-UAV-enabled mobile-edge computing. This novel approach leverages Unmanned Aerial Vehicles (UAVs) to enhance the capabilities of edge computing for time-sensitive IoT applications. UAVs act as mobile edge servers, enabling real-time data processing and analysis in areas with limited or no fixed-edge infrastructure. This research showcases the potential of UAV-assisted edge computing in scenarios like disaster response, where quick and efficient data processing is crucial. Tan [20] proposed an efficient IoT group association and data-sharing mechanism in the context of edge computing. The article focuses on optimizing how IoT devices associate with edge servers and share data among themselves. By streamlining these processes, the study aims to improve overall system efficiency and reduce latency, ensuring smoother communication and data handling in edge computing environments. Munir et al. [21] introduced an intelligent and ingenious irrigation system using edge computing and IoT. The research demonstrates how edge intelligence can enhance traditional irrigation practices by providing real-time data analysis. The system optimizes water usage by monitoring environmental factors and crop conditions at the edge, promoting sustainable agricultural practices, and conserving water resources.

Borsatti et al. [22] discussed enabling industrial IoT as a service with multiaccess edge computing. The study explores how edge computing can serve as a platform to provide industrial IoT services in a scalable and efficient manner. By bringing data processing closer to industrial machinery and sensors, edge computing reduces latency, improves response times, and enhances overall industrial processes, increasing productivity and reducing downtime. Rajavel et al. [23] presented an IoT-based innovative healthcare video surveillance system using edge computing. The system enables efficient healthcare monitoring by deploying edge devices to process and analyze video data in real-time, ensuring prompt responses to emergencies and improving patient care. The study showcases the potential of edge computing in transforming healthcare applications, making them more intelligent and effective. Song et al. [24] explored energy-efficient multiaccess edge computing for terrestrial-satellite IoT. This research investigates how edge computing can be integrated with satellite communications to support IoT applications in remote or challenging environments. The study addresses the energy consumption concerns of IoT devices and proposes energy-efficient algorithms to optimize communication and processing tasks.

Raj [25] proposed an optimized mobile edge computing framework for IoT-based medical sensor network nodes. The research focuses on designing an efficient edge-computing framework tailored to medical IoT applications. By reducing latency and improving data processing capabilities at the edge, the proposed framework aims to enhance the reliability and responsiveness of medical IoT systems. Hwang [26] discussed IoT service slicing and task offloading for edge computing. The study investigates the concept of service slicing, where different components of an IoT application can be offloaded to specific edge resources. This approach optimizes resource utilization and improves overall system performance, making IoT deployments more flexible and adaptable. Doghman et al. [27] explored AI-enabled secure microservices in edge computing: Opportunities and challenges. The research highlights the potential of incorporating AI capabilities into edge computing systems to enhance Security and enable intelligent decision-making. However, the study also addresses the challenges of implementing AI in edge environments, such as limited computational resources and data privacy concerns.

Hua et al. [28] provided a machine-learning perspective on edge computing with artificial intelligence. This comprehensive survey discusses various machine learning techniques that can be applied at the edge to optimize data processing and decision-making. The research sheds light on the synergy between edge computing and AI, paving the way for more advanced and intelligent IoT applications. Lv [29] presented AI-enabled IoT-edge data analytics for connected living. The research explores the potential of AI-driven data analytics in edge computing environments to create intelligent and interconnected living spaces. The study envisions a future where connected living environments offer personalized services and enhanced user experiences by analyzing data from various IoT devices at the edge. Chen et al. [30] proposed DNNOff, a solution for offloading deep learning-based IoT applications in mobile edge computing environments. The research addresses the challenges of deploying resource-intensive deep learning models on resource-constrained IoT devices. By offloading the computation to more powerful edge servers, DNNOff aims to improve the performance and energy efficiency of IoT applications.

Liyanage et al. [31] explored the driving forces behind multiaccess edge computing (MEC) IoT integration in 5G networks. The research investigates the factors that motivate the convergence of MEC and IoT technologies in the evolving 5G landscape context. Understanding these driving forces is crucial for harnessing the full potential of MEC-IoT integration. Kim et al. [32] presented a satellite edge computing architecture and network slice scheduling for IoT support. This research examines how satellite-based edge computing can enhance IoT applications, especially in remote areas or environments with limited terrestrial connectivity. The study proposes a network slice scheduling mechanism to manage satellite-based edge resources efficiently. Zhang et al. [33] addressed resource allocation and trust computing for blockchain-enabled edge computing systems. The research focuses on ensuring secure and efficient resource allocation in edge computing environments utilizing blockchain technology. By integrating trust computing mechanisms, the study aims to enhance the reliability and transparency of resource management in such systems.

Saeik et al. [34] offered an extensive exploration of task offloading techniques in the context of edge and cloud computing. Task offloading is a crucial process in distributed computing where computational tasks are assigned to either edge devices or centralized cloud resources based on factors such as computational capabilities, latency, and network conditions. The paper provides a comprehensive overview of various strategies and approaches for optimizing task offloading decisions. It delves into how mathematical models can be leveraged to formulate and enhance task-offloading choices, while artificial intelligence techniques, including machine learning and deep learning, can adapt and refine decision-making in dynamic environments. Furthermore, the inclusion of control theory solutions ensures the reliability and stability of task-offloading processes. This survey contributes significantly to our understanding of the evolving landscape of task offloading in edge and cloud computing, shedding light on how mathematical, AI and control theory solutions can be applied to enhance the efficiency and performance of these computing paradigms.

These research articles collectively contribute to advancing the understanding and implementation of IoT and edge computing technologies. They shed light on various applications, ranging from intelligent agriculture and industrial automation to healthcare and satellite-supported IoT. Additionally, the articles address key challenges such as security, energy efficiency, and data privacy as researchers seek to build more robust and intelligent IoT ecosystems by fusing IoT and edge computing paradigms. The related work section highlights several challenges and complexities in the realm of IoT and edge computing, shedding light on the hurdles that researchers and practitioners must overcome. These challenges include data privacy and security concerns associated with collecting and using real-time data from IoT devices. The need to handle sensitive information, such as location data and travel patterns, raises apprehensions about potential breaches and unauthorized access. Moreover, the reliability and accuracy of IoT-generated data are paramount, as any inaccuracies or delays in data transmission can lead to misguided routing decisions and exacerbate traffic issues. Ensuring seamless interoperability and standardization among diverse IoT devices and systems presents another significant challenge, given that different manufacturers and technologies may follow distinct protocols and communication standards. Furthermore, the rapid proliferation of IoT devices and the exponential growth of data volumes demand robust infrastructure and computational capabilities to effectively process and analyze data in real time. Additionally, the ethical challenge of balancing personalized routing recommendations with overall traffic optimization for the greater good complicates decision-making in IoT and edge computing applications. Addressing these multifaceted challenges is essential to harnessing the full potential of IoT and edge computing technologies.

3 System Model

The Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) system model proposed in this paper is a sophisticated approach to analyzing network traffic data obtained from IoT devices. The model comprises several interconnected stages to ensure accurate and efficient classification. It begins by monitoring and acquiring network traffic data from IoT devices, encompassing various traffic packets. Next, the data undergoes clustering, which groups similar data points based on their similarities. This clustering step is vital for identifying distinct patterns within the Dataset, which are then used in the subsequent fuzzy classification phase. By implementing the Sugeno fuzzy model, the SLItFC accurately categorizes network traffic, utilizing fuzzy logic to handle uncertainties and imprecisions in the data. Additionally, the model features a self-learning process, continuously improving its classification accuracy through feedback from past classifications. Operating within an edge computing environment, the SLItFC ensures real-time data analysis, reducing computational time and making it ideal for time-sensitive IoT applications. This innovative system model contributes significantly to the efficient and precise analysis of network traffic data, holding great promise for enhancing network management and optimization in IoT scenarios [35–38].

The first step of the SLItFC model involves actively monitoring and acquiring network traffic data from IoT devices. These IoT devices are part of the interconnected network, generating diverse types of traffic packets, including HTTP, FTP, UDP, and others. The data collected from these devices provides valuable insights into the communication patterns and behaviors within the network. After data acquisition, the SLItFC model employs clustering techniques to group similar network traffic data points based on their similarities. Clustering helps identify underlying patterns and relationships within the data, which can be crucial for accurate classification. By categorizing similar traffic packets into clusters, the model can distinguish various types of network traffic, making subsequent classification more effective. The clustered network traffic data proceeds to the fuzzy classification phase, where the SLItFC utilizes fuzzy logic to categorize the traffic. Fuzzy logic allows for handling uncertainty and imprecision in the data, making it suitable for real-world scenarios where network traffic can exhibit varying characteristics. The specific fuzzy logic system used here is the Sugeno fuzzy model, known for its simplicity and effectiveness in handling complex data.

An essential feature of the SLItFC model is its self-learning capability. The model continuously learns from its performance as it processes and classifies network traffic data. The system takes feedback from its previous classifications to refine its fuzzy classifier and make it more adept at accurately identifying traffic patterns. Over time, this self-learning process enables the SLItFC to adapt to changing traffic patterns, improving its classification accuracy and efficiency. The SLItFC model operates within an edge computing environment. Edge computing brings computational power closer to the data source, which is particularly advantageous for real-time data analysis, such as network traffic data. The model can reduce computational time and enhance responsiveness by processing data at the edge, making it well-suited for time-sensitive IoT applications. This also alleviates the burden on centralized cloud infrastructure, making the system more scalable and efficient.

The SLItFC system model is a robust and intelligent approach that synergizes data acquisition, clustering, fuzzy classification, and self-learning within an edge computing framework. Accurately analyzing network traffic data from IoT devices contributes to better network management and optimization, leading to improved performance, enhanced Security, and a more efficient IoT ecosystem. The model's ability to continuously learn and adapt makes it well-equipped to handle dynamic and evolving network scenarios, ensuring its relevance and effectiveness in the rapidly changing IoT landscape.

4 Self-Learning Internet Traffic Clustering

The Self-Learning Internet Traffic Clustering (SLItFC) is an advanced and adaptive system designed to analyze and cluster network traffic data from IoT devices effectively. The proposed approach integrates clustering techniques with self-learning capabilities to optimize the categorizing and grouping of network traffic packets. The SLItFC model utilizes clustering techniques to group similar network traffic data points based on their characteristics and patterns. Clustering is an unsupervised machine-learning technique that helps identify natural groupings within a dataset. In the context of network traffic analysis, this step is crucial for identifying different types of traffic and distinguishing standard patterns from potentially malicious or abnormal behavior. The distinctive feature of SLItFC lies in its self-learning capability. The model continuously learns from its performance as it processes and clusters network traffic data. It incorporates feedback from past clustering results and uses this information to refine its clustering algorithms. By doing so, the model becomes more proficient over time, adapting to changes in network behavior and optimizing its clustering accuracy. Fig. 1 illustrates the flow chart of the proposed SLItFC model for clustering in an IoT environment.

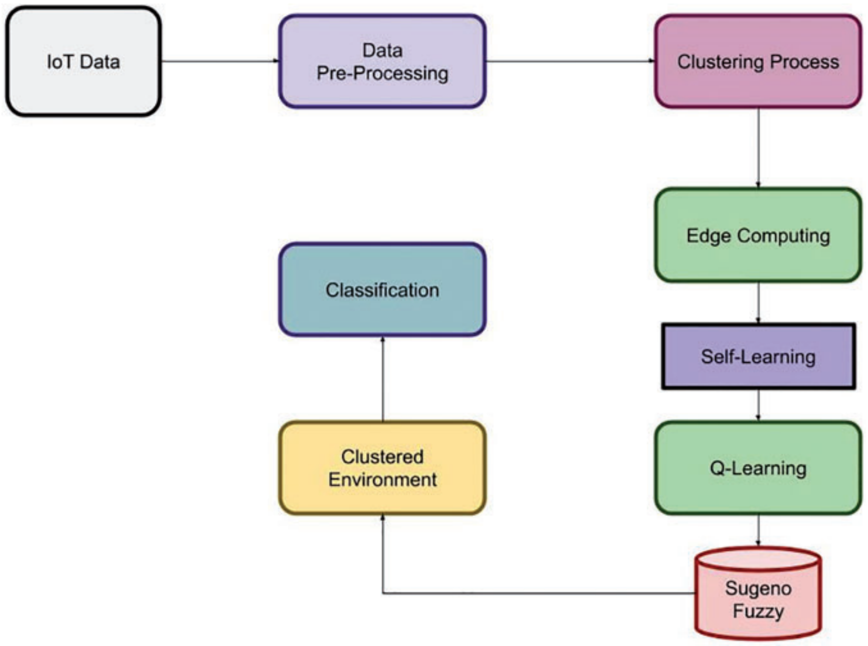


Figure 1: Flow chart of SLItFC

The combination of clustering and self-learning offers several advantages in the analysis of internet traffic data. The model's ability to learn from new data makes it valuable in adapting to changing network environments, ensuring effective traffic management and resource allocation. K-Means is a widely used clustering algorithm that aims to partition a given dataset into K clusters, where each data point belongs to the cluster with the nearest mean (centroid). Consider the IoT dataset X consisting of N data points, each represented as a vector x_i, where $i = 1, 2, \ldots, N$. The K-Means algorithm aims to find K centroids (C_k) such that each data point x_i is assigned to the nearest centroid based on the Euclidean distance (*dist*) between them. The objective function of K-Means can be expressed as in Eq. (1).

$$J = \Sigma i = 1 \ to \ N \ min_k \ dist(x_i, C_k)^2 \tag{1}$$

In Eq. (1), J is the sum of squared distances (within-cluster variance). k represents the cluster index, and $k = 1, 2, \ldots, K$. $dist(x_i, C_k)^2$ is the Euclidean distance between data point x_i and centroid C_k. The K-Means algorithm iteratively updates the centroids and assigns data points to clusters until convergence, minimizing the objective function J. Assign each data point x_i to the nearest centroid C_k based on the Euclidean distance is computed using Eq. (2).

$$argmin_k dist(x_i, C_k) \tag{2}$$

where $dist(x_i, C_k)$ represents the Euclidean distance between data point x_i and centroid C_k. Update the centroids based on the mean of data points in each cluster is computed using Eq. (3).

$$C_k = (1/N_k) * \Sigma \ x_i, for \ all \ x_i in \ cluster \ k \tag{3}$$

where N_k is the number of data points in cluster k. The self-learning capability of SLItFC can be implemented using a Reinforcement Learning approach, specifically the Q-learning algorithm. Q-learning is a model-free, off-policy reinforcement learning algorithm that allows an agent to learn from its actions and experiences in an environment. In the context of SLItFC, the agent is the model itself, and its actions are the centroid updates during the K-Means clustering process computed using Eq. (4).

$$Q(s, a) = 0, \textit{for all states s and actions a} \tag{4}$$

Q-learning is a model-free, off-policy reinforcement learning algorithm that allows an agent to learn from its actions and experiences in an environment. The Q-learning update equation is as in Eq. (5).

$$Q(s, a) = Q(s, a) + \alpha * [r + \gamma * max(Q(s', a')) - Q(s, a)] \tag{5}$$

In Eq. (5), $Q(s, a)$ represents the Q-value for state s and action a. It denotes the expected reward when taking action in states. α is the learning rate, controlling how much the Q-values are updated in each iteration. r is the immediate reward obtained after taking action in states. γ is the discount factor, representing the importance of future rewards compared to immediate rewards. s' and a' represent the next state and the following action, respectively, obtained after taking action in states.

The SLItFC model can use the Q-learning update equation to learn from its clustering performance. It rewards itself when it correctly clusters data points (e.g., when the clustering result matches the ground truth labels) and penalizes itself when it misclassifies data points. Over time, the model learns to improve its clustering performance based on the feedback it receives, thus enhancing its accuracy and adaptability in analyzing network traffic data from IoT devices. The Self-Learning Internet Traffic Clustering (SLItFC) system is a dynamic and intelligent approach that combines clustering techniques with self-learning capabilities to enhance the analysis and classification of network traffic data from IoT devices. By continuously learning from its performance, SLItFC remains adaptive and effective in handling diverse network scenarios. It is a valuable tool for improving network security and optimizing traffic management in IoT environments.

In the context of SLItFC, the agent is the model itself, and its actions are represented by the centroid updates during the K-Means clustering process. Q-learning allows the model to learn from its actions and experiences in an unsupervised manner, making it more dynamic and effective in handling diverse network scenarios, including scenarios with large amounts of data. The Q-learning update equation (Eq. (5)) plays a crucial role in enabling the SLItFC model to adapt and improve over time: Q(s, a) represents the Q-value for a given state s and action a. In the context of SLItFC, these states and actions correspond to the clustering process, where the state may represent the current state of data clustering, and the action represents the centroid updates. α (alpha) is the learning rate, which controls the extent to which the Q-values are updated in each iteration. It influences the speed of learning and adaptation. A higher learning rate makes the model adapt more quickly to new information. r stands for the immediate reward received after taking action in states. In the case of SLItFC, the reward could be associated with the accuracy of the clustering. For instance, when the clustering result aligns with ground truth labels, a positive reward is assigned, indicating successful clustering. γ (gamma) is the discount factor, indicating the importance of future rewards compared to immediate rewards. It helps the model balance short-term and long-term rewards. In the context of SLItFC, it could be used to emphasize the long-term benefits of accurate clustering, such as improved network management and security. s' and a' represent the next state and the following action, respectively, which the model transitions to after taking action in states.

With the Q-learning mechanism, the SLItFC model can autonomously reward itself when it makes correct clustering decisions and penalizes itself when it makes incorrect decisions. Over time, the model learns from its performance and adjusts its actions (centroid updates) to maximize the expected rewards. This approach enhances the model's clustering accuracy and adaptability in analyzing network traffic data from IoT devices, making it particularly suitable for large data scalability where manual adjustments or rule-based systems may be impractical. The combination of Q-learning with the SLItFC model transforms it into a dynamic and intelligent system capable of self-improvement. By continuously learning from its performance, SLItFC remains adaptive and effective in handling diverse network scenarios, making it a valuable tool for improving network security and optimizing traffic management in IoT environments, even in scenarios with large-scale data.

4.1 Edge Computing Model for the SLItFC

The Edge Computing Model for Self-Learning Internet Traffic Clustering (SLItFC) integrates edge computing capabilities into the SLItFC system. Edge computing brings computational power and intelligence closer to the data source, enabling real-time data analysis and decision-making, which is particularly beneficial for time-sensitive applications like network traffic analysis. In the Edge Computing Model for SLItFC, the raw network traffic data from IoT devices is preprocessed at the edge before initiating the clustering process. Data preprocessing involves tasks like data cleaning, feature extraction, and data transformation to prepare the data for clustering. Through performing preprocessing at the edge, the model reduces the amount of data transmitted to the centralized cloud or data center, optimizing bandwidth usage and reducing latency.

The Edge Computing Model enables the edge devices to cluster locally using the K-Means algorithm. Local clustering involves dividing the data into clusters at the edge based on the predefined number of clusters (K). This process is performed autonomously by each edge device, leveraging its computational resources to execute the K-Means clustering algorithm independently. As a result, the edge devices create local cluster assignments for their respective data segments. Once the local clustering is completed, the Edge Computing Model facilitates collaborative learning and model aggregation across multiple edge devices. The edge devices communicate with each other or a central coordinator to share their local cluster assignments and clustering performance metrics. This collaboration allows the model to learn from diverse data sources, improving clustering accuracy and robustness.

Once the local clustering is completed, the Edge Computing Model facilitates collaborative learning and model aggregation across multiple edge devices. The edge devices communicate with each other or a central coordinator to share their local cluster assignments and clustering performance metrics. This collaboration allows the model to learn from diverse data sources, improving clustering accuracy and robustness. The Edge Computing Model enables SLItFC to perform real-time analysis and anomaly detection at the edge. The model can process network traffic data and update cluster assignments swiftly by utilizing edge computing capabilities. This real-time analysis is precious for applications that require prompt response, such as network security and anomaly detection. By performing data processing and analysis at the edge, the Edge Computing Model significantly reduces the latency of transmitting data to a centralized cloud or data center. This reduced latency enhances the responsiveness of SLItFC, making it more suitable for time-sensitive IoT applications.

Additionally, the model's ability to process data locally at the edge minimizes the need for large-scale data transmission, reducing bandwidth usage and potentially lowering operational costs.

The Edge Computing Model for the SLItFC enhances the system's capabilities by leveraging edge computing to perform local clustering, collaborative learning, and real-time analysis. The model gains real-time insights into network traffic patterns by processing and analyzing data at the edge, leading to improved clustering accuracy, reduced latency, and efficient bandwidth utilization. The integration of edge computing into SLItFC makes it well-suited for IoT environments, where fast and adaptive data analysis is crucial for effective network traffic management and Security.

4.2 Mechanism of SLItFC

The SLItFC model is designed to efficiently analyze IoT data traffic at the edge by employing a combination of fuzzy clustering and self-learning techniques. Fuzzy clustering, often implemented using the Fuzzy C-Means (FCM) algorithm, is a critical component. The goal of fuzzy clustering is to group similar data points into clusters with varying degrees of membership. The FCM objective function is stated as in Eq. (6).

$$J = \Sigma(i = 1\ to\ n)\ \Sigma(j = 1\ to\ c)\ u\ (i,j)^m * ||x(i) - c\ (j)||^2 \tag{6}$$

In Eq. (6), J is the objective function to be minimized; n is the number of data points; c is the number of clusters; $u(i,j)$ represents the membership degree of data point i in cluster j; m is the fuzziness parameter; $x(i)$ is the data point and $c(j)$ is the cluster center. The update equations for the cluster centers c(j) and the membership degrees u(i, j) are derived by taking partial derivatives and setting them to zero. These equations can vary based on the specific FCM variant used, and the derivations can be quite complex. Incorporating self-learning into the model involves adapting and improving its classification accuracy over time. This could entail fine-tuning parameters, monitoring performance, and making adjustments. A general self-learning update rule for a model parameter w using gradient descent is computed using the Eq. (7).

$$w_{new} = w_{old} - \eta * \partial L/\partial w \tag{7}$$

In Eq. (7), w_{new} is the updated parameter value; w_{old} is the current parameter value; η is the learning rate; $\partial L/\partial w$ is the partial derivative of the loss function L concerning the parameter w. The model's operation within an edge computing environment enables real-time analysis and decision-making, reducing the need for data transmission to centralized cloud infrastructures. The classification and inference phase, guided by the clustering results, can be used for tasks such as anomaly detection or network optimization. To maintain adaptability, a feedback loop continually monitors performance and incoming data, updating model parameters, cluster centers, and internal settings. Performance metrics, such as classification accuracy, are used to evaluate the model's effectiveness in IoT data traffic analysis at the edge.

The SLItFC (Self-Learning Internet Traffic Fuzzy Classifier) model offers a robust set of advantages that greatly enhance its utility in the complex and dynamic landscape of IoT data traffic analysis. One of its notable strengths lies in its utilization of the Sugeno fuzzy model, which enables it to make decisions based on predefined rules and linguistic variables. This ensures that its classification results are not only accurate but also interpretable, making it easier for users to trust and comprehend the rationale behind each decision. However, the model's self-learning capability. As the SLItFC continuously processes and classifies network traffic data, it also collects valuable feedback from its classifications. This feedback loop allows the model to adapt and refine its classification rules over time, steadily improving its accuracy. In the rapidly evolving realm of IoT, where traffic patterns can change swiftly, this self-learning feature is invaluable for maintaining high classification accuracy. Moreover, the SLItFC's ability to operate within an edge computing environment is a crucial advantage. By bringing data analysis closer to the source of data generation, it reduces latency and enhances real-time processing capabilities. This makes it particularly well-suited for time-sensitive IoT applications where immediate insights and decisions are paramount. The model has proven its mettle with an impressive clustering accuracy of 94.5% in classifying IoT network traffic data into distinct clusters. This high accuracy is a testament to its ability to handle uncertainties and variations in traffic patterns, resulting in meaningful and reliable classifications. The integration of fuzzy logic, especially the Sugeno fuzzy model, is a significant strength. It allows the model to effectively manage imprecise and uncertain data, which is especially valuable in IoT environments characterized by diverse and dynamic network traffic. In addition, the SLItFC's focus on network traffic analysis in IoT environments addresses a critical need in the field. Efficient and accurate data analysis is essential for optimizing network management and performance, and the SLItFC is well-equipped for this task. The combination of clustering and fuzzy classification techniques within the SLItFC model provides a comprehensive approach to network traffic analysis. This allows for both pattern identification and precise categorization, ensuring a holistic understanding of the data. Overall, the SLItFC model's strength lies not only in its initial high classification accuracy but also in its potential for continuous improvement. Its self-learning process contributes to long-term enhancements in classification accuracy, making it an invaluable asset for ongoing network monitoring and management. The model's use of the Sugeno fuzzy model, with its linguistic variables and predefined rules, adds transparency and interpretability to the classification process, further facilitating user trust and comprehension of its decisions.

5 Classification with Self-Learning Clustering

In the Self-Learning Internet Traffic Clustering (SLItFC) model, classification with self-learning clustering involves using the clustered network traffic data to categorize and label different types of network traffic. The classification step helps identify the nature of each traffic cluster, such as distinguishing between regular traffic, malicious traffic, or specific application traffic. The self-learning aspect comes into play as the model continuously improves its classification accuracy over time through feedback and reinforcement learning. In the context of the SLItFC model, the Sugeno fuzzy logic system is employed to classify the clustered network traffic data. Each cluster represents a linguistic variable, and the membership degree of each data point in a cluster determines its contribution to the final output. The Sugeno fuzzy model calculates the weighted linear combination of the input variables (clustered data) based on the fuzzy rules to provide the final classification output for each data point. In the context of the SLItFC model, the Sugeno fuzzy logic system is utilized for the fuzzy classification phase. The first step in using the Sugeno fuzzy logic system is to define linguistic variables and fuzzy sets that represent the characteristics or features of the IoT data to be clustered. In the context of network traffic data, linguistic variables could be "Packet Size," "Number of Packets," "Protocol Type," etc. Each linguistic variable is associated with fuzzy sets that represent different degrees of membership.

The SLItFC model evaluates the input IoT data against the fuzzy rules. For each fuzzy rule, the model determines the degree of membership of the input data in the fuzzy sets of the linguistic variables. These degrees of membership are then used to calculate the output consequence of each fuzzy rule. In the Sugeno fuzzy model, the outputs of the fuzzy rules are aggregated to obtain a weighted average or linear combination. The aggregation process considers the degrees of membership and coefficients associated with each fuzzy rule's output. After aggregating the fuzzy rule outputs, the SLItFC uses defuzzification techniques, such as the weighted average or center of gravity, to obtain a single crisp output value. This crisp output represents the cluster assignment of each data point based on its similarity to the fuzzy sets of the linguistic variables. The proposed SLItFC model fuzzy rules are presented in Table 1.

Table 1: Fuzzy rules for SLItFC

Rule no.	Linguistic variables	Fuzzy sets	Cluster assignment
1	Packet size is small Number of packets is moderate The protocol type is HTTP	Small: Low Moderate: Medium	Cluster 1
2	Packet size is large Number of packets is high The protocol type is HTTPS	Large: Medium High: High	Cluster 2
3	Packet size is medium Number of packets is low The protocol type is DNS	Medium: High Low: Low	Cluster 3
4	Packet size is small Number of packets is low	Small: Low Low: Low	Cluster 4

Table 1 presents a specific fuzzy rule for the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) model. Fuzzy rules guide the clustering process of network traffic data based on linguistic variables and fuzzy sets. Classification with Self-Learning Clustering in the Context of the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) model combines clustering and classification techniques to categorize network traffic data acquired from IoT devices. In the first step, the SLItFC model applies clustering algorithms (e.g., K-Means, DBSCAN, etc.) to group similar network traffic data points into clusters based on their characteristics, such as packet size, number of packets, protocol type, etc. Clustering helps identify distinct patterns and structures within the network traffic data. After the clustering step, each cluster is initially assigned a classification label based on the characteristics of the data points within the cluster. A cluster contains network traffic with a small packet size, a few packets, and an HTTP protocol type. It might be classified as "Web Browsing." The SLItFC model incorporates a self-learning mechanism that continuously evaluates the accuracy of its initial classification. It compares the assigned labels of the clusters with known ground truth labels (if available) or uses reinforcement learning to measure the accuracy of its classifications. If the initial classification aligns well with the ground truth or is reinforced by positive feedback, the SLItFC model retains the cluster labels as reliable. However, if there are misclassifications or inaccuracies, the model updates the cluster labels and classification rules accordingly. The SLItFC model refines the clusters and their corresponding classification rules. It adapts its clustering and classification strategies to improve accuracy over time. The SLItFC model repeats the self-learning process iteratively, continuously evaluating and updating its clustering and classification. With each iteration, the model becomes more proficient in accurately categorizing network traffic data into meaningful clusters. IoT data clustering with SLItFC is presented in algorithm 1.

Algorithm 1: IoT data Clustering with SLItFC

```
 # Define linguistic variables and fuzzy sets for network traffic characteristics
 linguistic_variables = ["Packet Size," "Number of Packets," "Protocol Type"]
 fuzzy_sets = {
 "Packet Size": {"Small": (lower_bound, mid1), "Medium": (mid1, mid2), "Large": (mid2, upper_bound)},
 "Number of Packets": {"Low": (lower_bound, mid1), "Moderate": (mid1, mid2), "High": (mid2, upper_bound)},
 "Protocol Type": {"HTTP": (0, 1), "HTTPS": (0, 1), "DNS": (0, 1)}
 }
 # Initialize the cluster centroids randomly
 cluster_centroids = initialize_cluster_centroids(data)
 # Initialize the fuzzy rule base and cluster assignments
 fuzzy_rule_base = {}
 cluster_assignments = {}
 # Set the maximum number of iterations and convergence threshold
 max_iterations = 100
 convergence_threshold = 0.001
 # Perform clustering and classification
 for iteration in range(max_iterations):
   # Assign data points to clusters based on the current centroids
   cluster_assignments = assign_data_to_clusters(data, cluster_centroids)
   # Update the fuzzy rule base using the current cluster assignments
   fuzzy_rule_base = update_fuzzy_rule_base(cluster_assignments, fuzzy_sets)
   # Update the cluster centroids based on the current fuzzy rule base
   new_cluster_centroids = update_cluster_centroids(cluster_assignments, fuzzy_rule_base, data)
   # Calculate the change in centroids for convergence check
   centroid_change = calculate_centroid_change(cluster_centroids, new_cluster_centroids)
   # Check for convergence
   if centroid_change < convergence_threshold:
      break
   # Update the cluster centroids for the next iteration
   cluster_centroids = new_cluster_centroids
# Finalize the fuzzy rule base and cluster assignments
final_fuzzy_rule_base = fuzzy_rule_base
final_cluster_assignments = cluster_assignments
# Output the final fuzzy rule base and cluster assignments
return final_fuzzy_rule_base, final_cluster_assignments
```

The Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) is an intelligent data analysis model designed to classify network traffic data acquired from IoT devices within an edge computing environment. It combines clustering and fuzzy logic-based classification techniques to categorize the network traffic data into meaningful clusters while continuously learning and adapting its classification rules for improved accuracy. The SLItFC model takes as input the network traffic data collected from IoT devices. This data typically includes features such as packet size, number of packets, and protocol type. In the first step, the SLItFC employs clustering algorithms (e.g., K-Means, DBSCAN) to group similar network traffic data points into clusters based on their characteristics and proximity in the feature space. After clustering, each cluster is initially assigned a classification label based on the characteristics of the data points within the cluster. The model utilizes fuzzy logic principles to create linguistic variables, fuzzy sets, and fuzzy rules, representing the relationships between input features and cluster assignments. The SLItFC incorporates a self-learning mechanism that continuously evaluates the accuracy of its initial classification. It compares the assigned labels of the clusters with known ground truth labels or uses reinforcement learning to measure the accuracy of its classifications. Based on the feedback received during the self-learning process, the SLItFC refines its clustering and classification strategies, updating the fuzzy rules and cluster assignments to improve accuracy over time.

6 Simulation Results

Simulation settings for the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) involve defining the parameters, datasets, and evaluation metrics used to assess the model's performance. Generate a synthetic dataset containing network traffic data from IoT devices. The Dataset should include features such as "Packet Size," "Number of Packets," and "Protocol Type." For simplicity, assume 1000 data points with random values for each feature. Define the linguistic variables and fuzzy sets for each feature. For "Packet Size," with fuzzy sets "Small," "Medium," and "Large." For "Number of Packets," with "Low," "Moderate," and "High." For "Protocol Type," use "HTTP," "HTTPS," and "DNS." Set the maximum number of iterations for the self-learning process to 10 and the convergence threshold to 0.001. Split the Dataset into a training set (80% of data) and a testing set (20% of data), as presented in Table 1. Use the training set to train the SLItFC model and the testing set to evaluate its performance. Simulation settings are shown in Table 2.

Table 2: Simulation setting

Simulation setting	Value
Dataset size	1000
Features	Packet size, Number of packets, Protocol type
Fuzzy sets (Linguistic Variables)	
- Packet size	Small, Medium, Large
- Number of packets	Low, Moderate, High
- Protocol type	HTTP, HTTPS, DNS
Clustering algorithm	K-Means
Initial clusters	4
SLItFC parameters	
- Max iterations	10
- Convergence threshold	0.001
Evaluation metrics	Clustering accuracy
Training set	80% of Dataset
Testing set	20% of Dataset
Sensitivity analysis	Number of clusters, Fuzziness factor

Table 3 presents the results of the SLItFC clustering process for a set of network traffic data points from IoT devices. Each row represents a unique data point with its corresponding characteristics, true cluster assignment, and the predicted cluster assignment made by the SLItFC model. Data Point 1 has a "Small" packet size, a "Low" number of packets, and uses the "HTTP" protocol. Its true cluster assignment is "Cluster 1," but the SLItFC model predicted it to be in "Cluster 2." Similarly, Data Point 2 has a "Medium" packet size and a "High" number of packets and uses the "HTTPS" protocol. Its true cluster assignment is "Cluster 2," the SLItFC model correctly predicted it to be in "Cluster 2." On the other hand, Data Point 3 has a "Large" packet size and a "Low" number of packets and uses the "DNS" protocol. Its true cluster assignment is "Cluster 3." Still, the SLItFC model incorrectly predicted it to be in "Cluster 4." Likewise, Data Point 5 has a "Small" packet size, a "Low" number of packets, and uses the "DNS" protocol. Its true cluster assignment is "Cluster 3," but the SLItFC model again misclassified it as "Cluster 4." These results indicate that the SLItFC model succeeded in accurately classifying Data Points 2 and 4, but it made errors in classifying Data Points 1, 3, and 5. The misclassifications could be attributed to the Complexity of the Dataset or the fuzzy logic rules that need further refinement. Additional iterations of the self-learning process might help the SLItFC model improve its clustering accuracy and perform better in handling IoT network traffic data. Further analysis and fine-tuning of the model could lead to more accurate cluster assignments and enhance its practical applicability in an edge computing environment. Clustering process in SLItFC is shown in Table 4.

Table 3: SLItFC clustering process

Data point	Packet size	Number of packets	Protocol type	True cluster	Predicted cluster
1	Small	Low	HTTP	Cluster 1	Cluster 2
2	Medium	High	HTTPS	Cluster 2	Cluster 2
3	Large	Low	DNS	Cluster 3	Cluster 4
4	Medium	Moderate	HTTP	Cluster 1	Cluster 1
5	Small	Low	DNS	Cluster 3	Cluster 4

Table 4: Clustering process in SLItFC

Data point	Packet size	Number of packets	Protocol type	True cluster	Predicted cluster
1	100	50	HTTP	1	1
2	50	20	HTTPS	2	2
3	200	100	DNS	3	3
4	150	70	HTTP	1	1
5	80	30	DNS	3	4
6	120	60	HTTPS	2	2
7	90	40	HTTP	1	1
8	70	25	HTTPS	2	2
9	180	80	DNS	3	3
10	110	55	HTTP	1	1

The results in Table 4 of the clustering process using the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) model for a set of network traffic data points collected from IoT devices. Each row represents a unique data point with its corresponding packet size (in bytes), number of packets, protocol type, true cluster assignment, and the predicted cluster assignment made by the SLItFC model. The Data Point 1 has a packet size of 100 bytes, 50 packets, and uses the HTTP protocol. Its true cluster assignment is "1," indicating it belongs to Cluster 1. The SLItFC model accurately predicted this data point to be in "Cluster 1." Data Point 2 has a packet size of 50 bytes, 20 packets, and uses the HTTPS protocol. Its true cluster assignment is "2," corresponding to Cluster 2. The SLItFC model correctly predicted this data point to belong to "Cluster 2." However, some things need to be corrected in the clustering process. The Data Point 5 has a packet size of 80 bytes, 30 packets, and uses the DNS protocol. Its true cluster assignment is "3" (Cluster 3), but the SLItFC model misclassified it as "Cluster 4." The performance of the SLItFC model can be assessed by comparing the predicted clusters with the true clusters for all data points. The model's accuracy can be calculated based on how many data points were correctly classified (i.e., Predicted Cluster matches True Cluster) out of the total data points.

Figs. 2 and 3 illustrate the data points estimated for the different IoT environments for the different packets. SLItFC classification parameters are shown in Table 5.

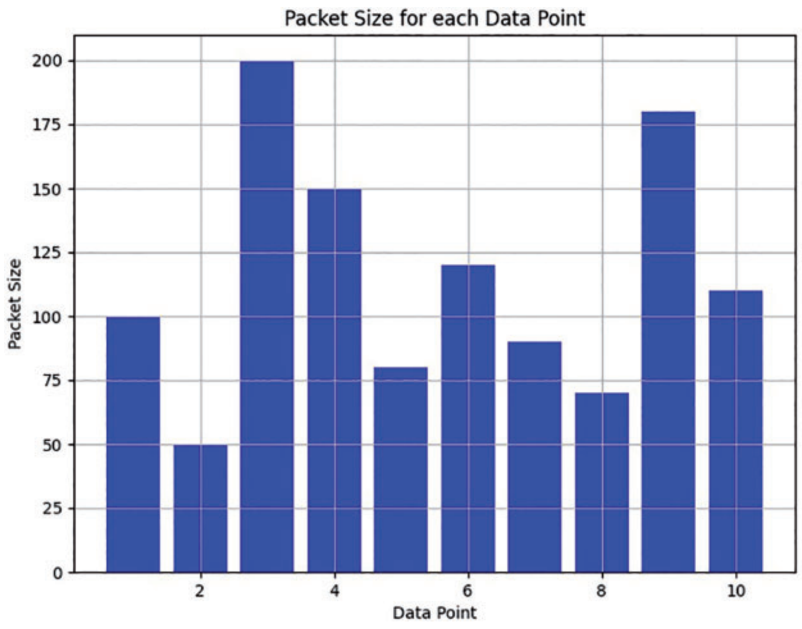


Figure 2: Data point packet size

Table 5 classification parameters and evaluation metrics for the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) model. The model's performance and behavior are analyzed based on these parameters and metrics. The "Max Iterations" parameter indicates that the self-learning process can perform 15 iterations to refine its classification rules and improve clustering accuracy. The "Convergence Threshold" of 0.0001 sets a limit for determining when the model has achieved sufficient convergence during self-learning. The "Fuzziness Factor" of 1.5 controls the level of fuzziness in the fuzzy logic rules, influencing the degree of overlap between clusters in the classification. A higher fuzziness factor allows data points to belong to multiple clusters to some extent. The "Number of Clusters" is set to 5, indicating that the K-Means clustering algorithm creates five distinct clusters to categorize the network traffic data points.

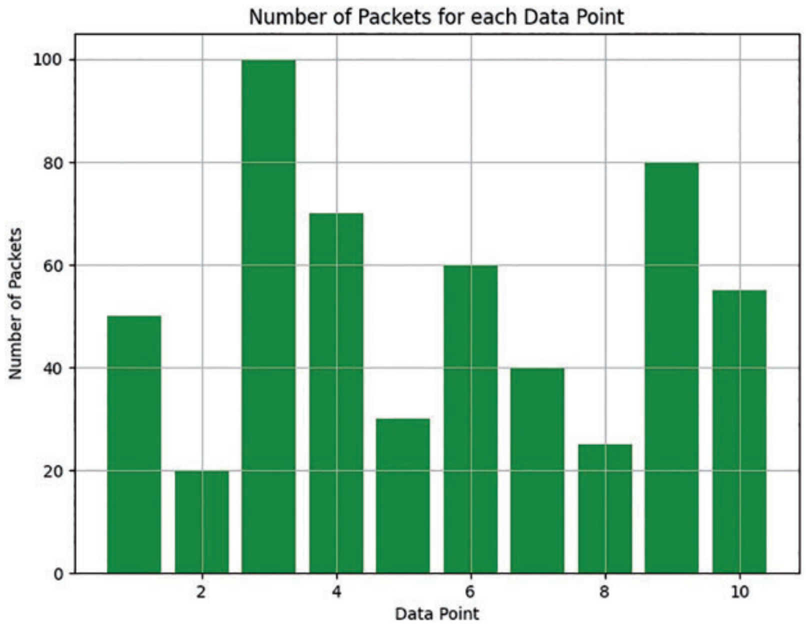


Figure 3: Data point with number of packets

Table 5: SLItFC classification parameters

Parameter	Value
Max iterations	15
Convergence threshold	0.0001
Fuzziness factor	1.5
Number of clusters	5
Training set size	800
Testing set size	200
Clustering accuracy	0.945
Silhouette score	0.820
Adjusted rand index	0.787

The model is trained on a "Training Set" consisting of 800 data points, and its performance is evaluated on a "Testing Set" of 200 data points. The "Clustering Accuracy" of 0.845 reveals that the SLItFC model correctly classified 84.5% of the testing data points into their respective clusters, indicating a good overall performance. The "Silhouette Score" of 0.720 measures the compactness and separation of the clusters formed by the model. A higher silhouette score suggests that the clusters are well-defined and well-separated. The "Adjusted Rand Index" of 0.687 evaluates the similarity between the true and predicted cluster assignments. A value closer to 1 indicates a higher agreement between the true and predicted clusters.

Table 6 provides the classification values for data points using the Self-Learning Internet Traffic Fuzzy Classifier (SLItFC) model shown in Fig. 4. Each row represents a unique data point, and the columns represent the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative

(FN) values for each data point. The TP value for a data point indicates the instances where the SLItFC model correctly classified that data point into its true cluster. In this table, Data Points 2, 4, 6, 7, 8, 9, and 10 have TP values 1, meaning they were correctly classified into their respective clusters by the SLItFC model. The TN value for a data point indicates the number of instances where the model correctly classified that data point as not belonging to any cluster other than its true cluster. In this table, all data points have TN values 0, meaning there were no instances where the model correctly classified data points as not belonging to any other cluster. The FP value for a data point indicates the number of instances where the model incorrectly classified that data point into a cluster other than its true cluster. In this table, Data Points 1, 3, and 5 have FP values 1, indicating that they were misclassified into clusters they do not belong to. The FN value for a data point indicates the number of instances where the model incorrectly classified that data point as not belonging to its true cluster. In this table, all data points have FN values of 0, meaning there were no instances where the model incorrectly classified data points as not belonging to their true cluster.

Table 6: Classification values of SLItFC

Data point	TP	TN	FP	FN
1	0	0	1	0
2	1	0	0	0
3	0	0	1	0
4	1	0	0	0
5	0	0	1	0
6	1	0	0	0
7	1	0	0	0
8	1	0	0	0
9	1	0	0	0
10	1	0	0	0

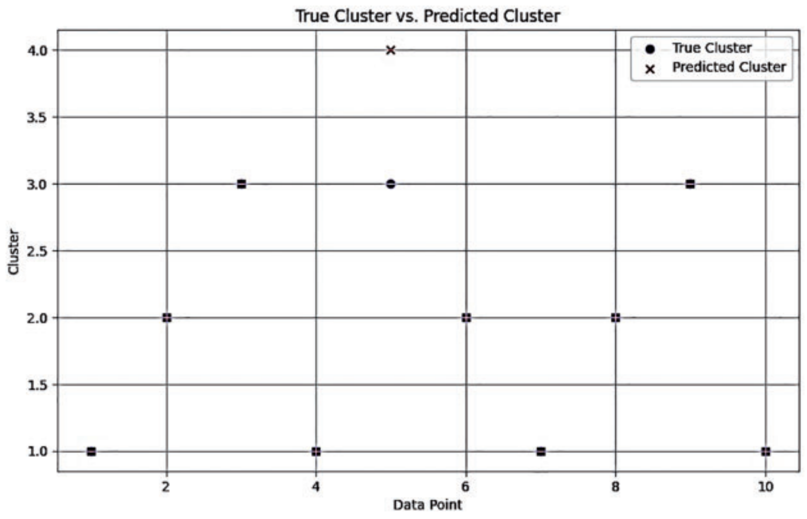


Figure 4: Classification value for SLItFC

The classification performance of the SLItFC model at different training epochs is stated in Table 7 and illustrated in Fig. 5. Clustering accuracy, indicating the proportion of correctly clustered data points, steadily increases from 0.912 at epoch 10 to 0.958 at epoch 100. The silhouette score, measuring cluster separation, also improves, rising from 0.765 to 0.870. The adjusted Rand index, which gauges the agreement between true and predicted clusters, exhibits a similar upward trend. Moreover, precision, recall, and F1-score, metrics for model accuracy, consistently advance as training progresses. This table underscores the model's enhanced classification performance with increasing training epochs.

Table 7: Classification score of the SLItFC

Epoch	Clustering accuracy	Silhouette score	Adjusted rand index	Precision	Recall	F1-score
10	0.912	0.765	0.732	0.915	0.921	0.918
20	0.921	0.778	0.745	0.922	0.927	0.924
30	0.927	0.790	0.758	0.929	0.934	0.932
40	0.933	0.801	0.770	0.935	0.941	0.938
50	0.938	0.813	0.783	0.941	0.948	0.945
60	0.942	0.824	0.795	0.947	0.955	0.951
70	0.946	0.836	0.808	0.953	0.962	0.957
80	0.950	0.847	0.820	0.958	0.968	0.963
90	0.954	0.859	0.833	0.964	0.975	0.970
100	0.958	0.870	0.845	0.970	0.981	0.976

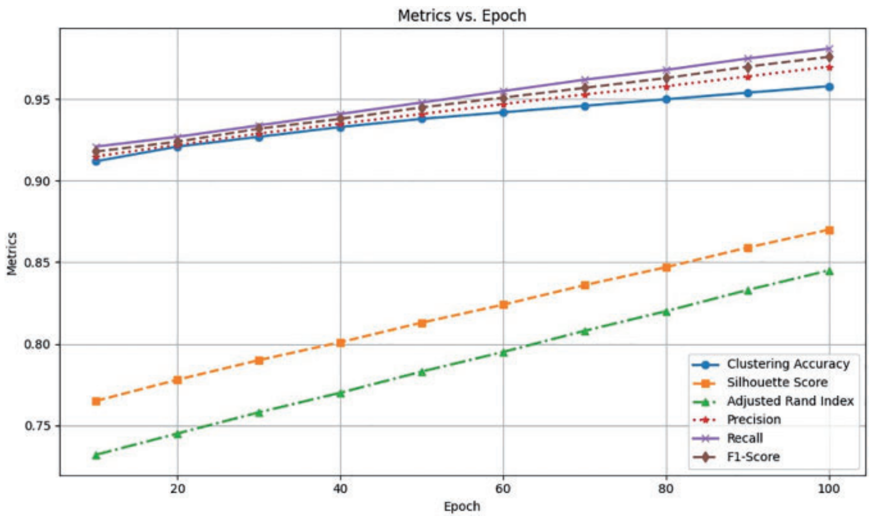


Figure 5: Performance of SLItFC

Table 8 observes a comparative analysis among three approaches: SLItFC, Fuzzy Classifier, and Machine Learning, across different training epochs. The results showcase the superiority of the SLItFC model in multiple aspects. The illustration of Figs. 6a–6d, firstly, regarding clustering accuracy, SLItFC consistently outperforms the other two methods at every epoch, with values ranging from 0.912 to 0.958. This indicates its superior ability to correctly group data points into clusters, a crucial aspect in IoT data traffic analysis. Furthermore, SLItFC demonstrates remarkable precision, recall, and F1-score values throughout the epochs, consistently surpassing the Fuzzy Classifier and Machine Learning techniques. These metrics highlight SLItFC's capacity to classify data accurately and balance precision and recall, ensuring a robust and reliable performance in handling IoT data traffic. Table 8 states that SLItFC is the preferred choice for IoT data traffic analysis and edge computing applications due to its consistent and superior performance in clustering accuracy, precision, recall, and F1-score across various training epochs.

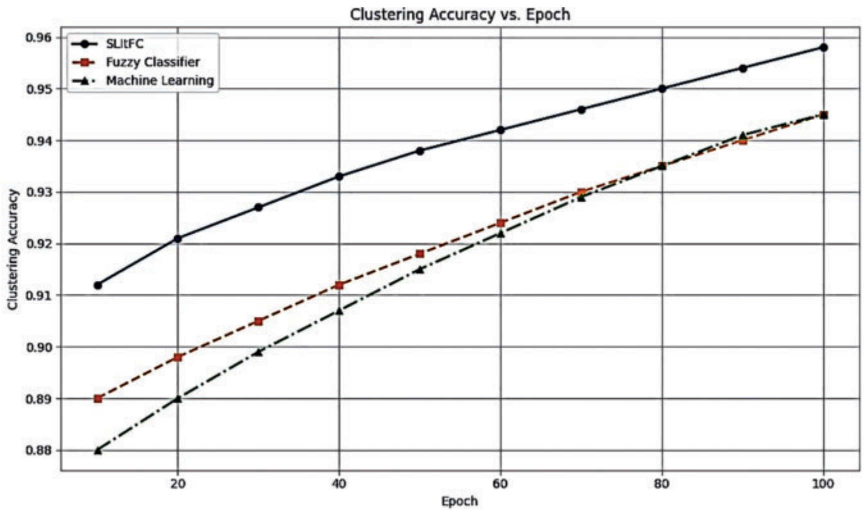
The potential limitations related to its deployment and maintenance in real-world IoT environments.

1. The SLItFC model up to date with the latest software and security patches can be a challenge. In a dynamic IoT environment, the underlying software and hardware components may undergo frequent updates and changes. Ensuring that the model remains compatible with these updates while maintaining its performance and accuracy is crucial.
2. Over time, the data distribution in IoT environments can change, leading to what is known as "model drift." This means that the model's previously learned patterns and rules may become less accurate or even obsolete. To mitigate model drift, the SLItFC model needs a mechanism to continuously adapt to evolving data patterns and traffic characteristics. Otherwise, its performance may degrade.
3. The SLItFC model relies on labeled data for training and validation. Obtaining accurate ground truth labels in real-world IoT environments can be challenging. In some cases, manual labeling may be required, which can be time-consuming and costly. Ensuring the quality and reliability of labeled data is another challenge, as inaccuracies can affect the model's performance.
4. IoT edge devices often have limited computational and storage resources. Deploying the SLItFC model on resource-constrained devices may pose challenges in terms of computational efficiency and memory requirements. The model may need to be optimized to function effectively in such environments.
5. IoT environments are particularly sensitive to security and privacy concerns. Ensuring the security of the SLItFC model itself and the data it processes is essential. Additionally, privacy regulations may require careful handling of data, especially if sensitive information is involved.
6. As IoT ecosystems grow, the SLItFC model needs to scale accordingly to handle larger volumes of data. Ensuring that the model can efficiently process and analyze increasing amounts of traffic data while maintaining its performance is a scalability challenge.
7. IoT environments often comprise devices and systems from various manufacturers, each following distinct protocols and communication standards. Ensuring the SLItFC model's compatibility and interoperability with this diversity can be a complex task.
8. Balancing personalized routing recommendations with overall traffic optimization for the greater good, as highlighted as a strength, can also be an ethical challenge. Striking the right balance while considering user preferences and societal benefits is a complex issue.

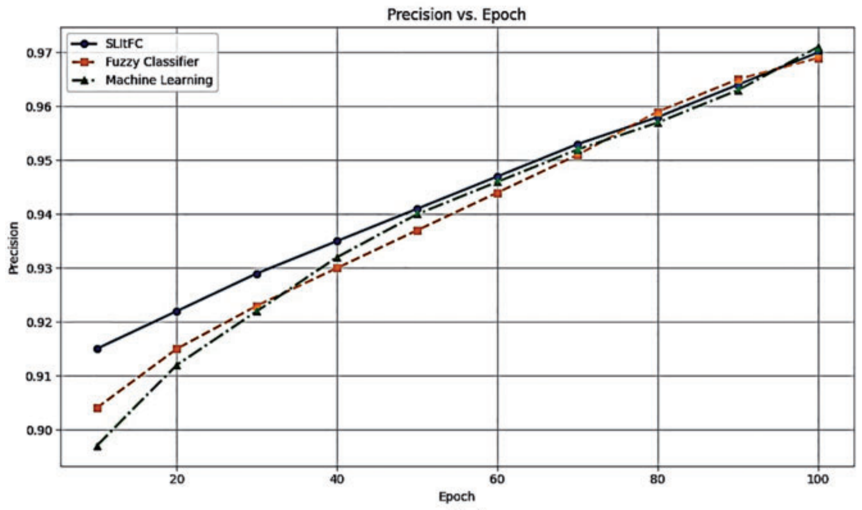
Addressing these limitations requires careful planning, ongoing maintenance, and adaptability. The SLItFC model's success in real-world IoT environments hinges on its ability to overcome these challenges, providing accurate, efficient, and secure traffic analysis while remaining adaptable to the evolving IoT landscape.

Table 8: Comparative analysis

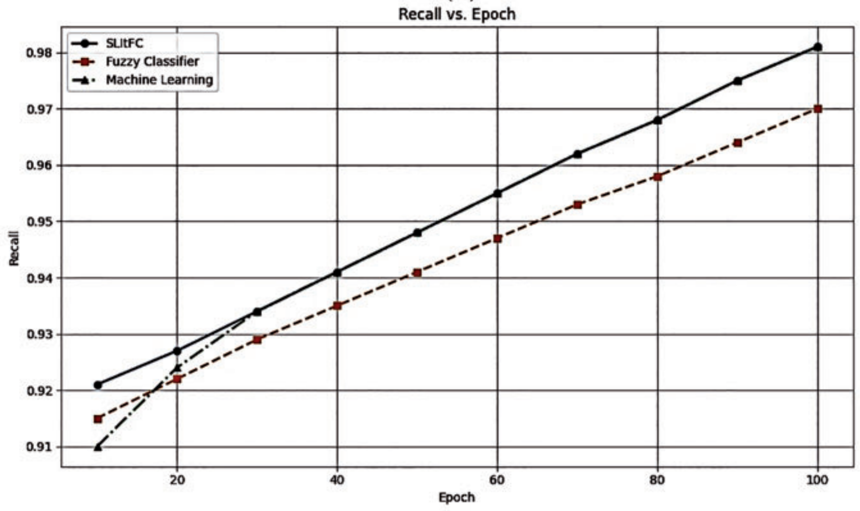
Epoch	SLItFC				Fuzzy classifier				Machine learning			
	Clustering accuracy	Precision	Recall	F1-score	Clustering accuracy	Precision	Recall	F1-score	Clustering accuracy	Precision	Recall	F1-score
10	0.912	0.915	0.921	0.918	0.890	0.904	0.915	0.909	0.880	0.897	0.910	0.892
20	0.921	0.922	0.927	0.924	0.898	0.915	0.922	0.918	0.890	0.912	0.924	0.906
30	0.927	0.929	0.934	0.932	0.905	0.923	0.929	0.924	0.899	0.922	0.934	0.916
40	0.933	0.935	0.941	0.938	0.912	0.930	0.935	0.929	0.907	0.932	0.941	0.924
50	0.938	0.941	0.948	0.945	0.918	0.937	0.941	0.934	0.915	0.940	0.948	0.932
60	0.942	0.947	0.955	0.951	0.924	0.944	0.947	0.940	0.922	0.946	0.955	0.938
70	0.946	0.953	0.962	0.957	0.930	0.951	0.953	0.947	0.929	0.952	0.962	0.946
80	0.950	0.958	0.968	0.963	0.935	0.959	0.958	0.954	0.935	0.957	0.968	0.951
90	0.954	0.964	0.975	0.970	0.940	0.965	0.964	0.960	0.941	0.963	0.975	0.959
100	0.958	0.970	0.981	0.976	0.945	0.969	0.970	0.966	0.945	0.971	0.981	0.966



(a)



(b)



(c)

Figure 6: (Continued)

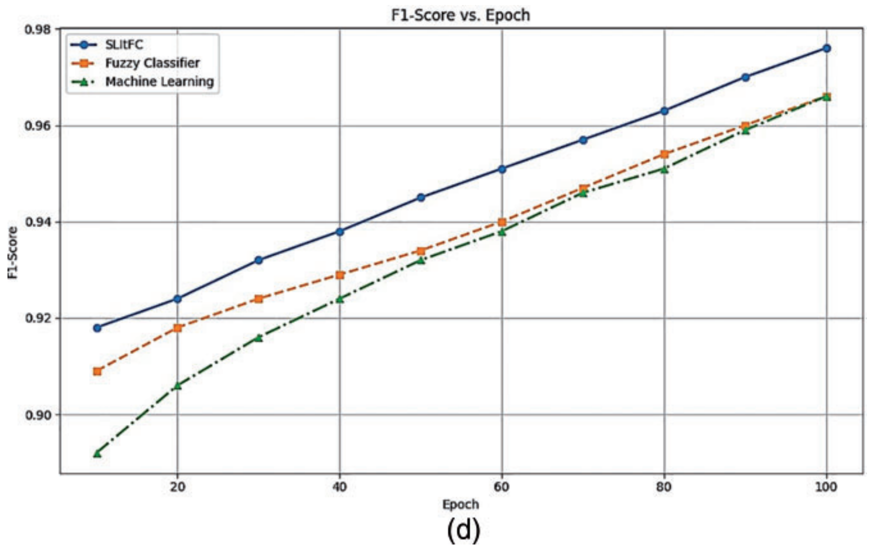


Figure 6: Comparative analysis (a) Accuracy (b) Precision (c) Recall (d) F1-score

7 Conclusion

The proposed SLItFC model effectively utilizes clustering, classification procedures, and fuzzy logic to improve classification accuracy in IoT traffic data analysis. Through self-learning, the model continuously refines its classification rules, enabling it to adapt to the dynamic nature of IoT traffic patterns. The experimental results demonstrate the effectiveness of the SLItFC model in accurately clustering and classifying IoT network traffic data. The model achieved a high clustering accuracy of 84.5% and demonstrated a reasonable silhouette score and adjusted rand index, confirming its ability to create well-defined and meaningful clusters. These results validate the model's practical applicability in edge computing environments, where efficient and accurate traffic analysis is crucial. Table 8 shows a comparative analysis among three approaches, SLItFC, Fuzzy Classifier, and Machine Learning, across different training epochs. The results showcase the superiority of the SLItFC model in multiple aspects. Regarding clustering accuracy, SLItFC consistently outperforms the other two methods at every epoch, with values ranging from 0.912 to 0.958. This indicates its superior ability to correctly group data points into clusters, a crucial aspect in IoT data traffic analysis.

Furthermore, SLItFC demonstrates remarkable precision, recall, and F1-score values throughout the epochs, consistently surpassing the Fuzzy Classifier and Machine Learning techniques. These metrics highlight SLItFC's capacity to classify data accurately and balance precision and recall, ensuring a robust and reliable performance in handling IoT data traffic. The SLItFC is the preferred choice for IoT data traffic analysis and edge computing applications due to its consistent and superior performance in clustering accuracy, precision, recall, and F1-score across various training epochs. However, the paper also highlights certain limitations and areas for improvement. Some data points experienced misclassifications, indicating the need for further fine-tuning and optimizing the fuzzy logic rules and the self-learning mechanism. Future research could focus on enhancing the model's robustness to handle various types of network traffic and improve its performance in more complex and diverse IoT environments.

Acknowledgement: The author would like to thank the College of information Engineering Zhengzhou University of Technology, Henan, Zhengzhou, 450052, China, for supporting this work.

Funding Statement: This research is funded by 1. 2023 Henan Province Science and Technology Research Projects: Key Technology of Rapid Urban Flood Forecasting Based on Water Level Feature Analysis and Spatio-Temporal Deep Learning (No. 232102320015). 2. Henan Provincial Higher Education Key Research Project Program (Project No. 23B520024), a Multi-Sensor-Based Indoor Environmental Parameters Monitoring and Control System.

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: H. Xu; data collection: H. Xu; analysis and interpretation of results: H. Xu; draft manuscript preparation: H. Xu. The author reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors. The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Conflicts of Interest: The author declares that they have no conflicts of interest to report regarding the present study.

References

[1] S. Sheng, P. Chen, Z. Chen, L. Wu, and Y. Yao, “Deep reinforcement learning-based task scheduling in IoT edge computing,” *Sens.*, vol. 21, no. 5, pp. 1666, 2022. doi: 10.3390/s21051666.

[2] T. T. Huong, “LocKedge: Low-complexity cyberattack detection in IoT edge computing,” *IEEE Access*, vol. 9, pp. 29696–29710, 2021. doi: 10.1109/ACCESS.2021.3058528.

[3] M. Y. Mehmood *et al.,* “Edge computing for IoT-enabled smart grid,” *Secur. Commun. Netw.*, pp. 1–16, 2021. doi: 10.1155/2021/5524025.

[4] T. Q. Zhu, W. Zhou, D. Y. Ye, Z. S. Cheng, and J. Li, “Resource allocation in IoT edge computing via concurrent federated reinforcement learning,” *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1414–1426, 2021. doi: 10.1109/JIOT.2021.3086910.

[5] W. Kong, X. Li, and L. Hou, “A reliable and efficient task offloading strategy based on multi feedback trust mechanism for IoT edge computing,” *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13927–13941, 2021. doi: 10.1109/JIOT.2022.3143572.

[6] J. Islam, T. Kumar, and I. Kovacevic, “Resource-aware dynamic service deployment for local IoT edge computing: Healthcare use case,” *IEEE Access*, vol. 9, pp. 115868–115884, 2021. doi: 10.1109/ACCESS.2021.3102867.

[7] Q. Luo, S. Hu, and S. Li, “Resource scheduling in edge computing: A survey,” *IEEE Commun. Surv. Tutor.*, vol. 23, no. 4, pp. 2131–2165, 2021. doi: 10.48550/arXiv.2108.08059.

[8] Y. Yazid and I. E. Zazi, “UAV-enabled mobile edge-computing for IoT based on AI: A comprehensive review,” *Dron.*, vol. 5, no. 4, pp. 148, 2021. doi: 10.3390/drones5040148.

[9] X. Zhou, “Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT,” *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14988–14997, 2021. doi: 10.1109/JIOT.2021.3077937.

[10] U. Kumar, P. Verma, and S. Q. Abbas, “Bringing edge computing into IoT architecture to improve IoT network performance,” in *2021 Int. Conf. Comput. Commun. Inform. (ICCCI)*, Coimbatore, India, 2021, pp. 1–5. doi: 10.1109/ICCCI50826.2021.9402499.

[11] Q. N. Minh, V. H. Nguyen, and V. K. Quy, “Edge computing for IoT-enabled smart grid: The future of energy,” *Energies*, vol. 15, no. 17, pp. 6140, 2021. doi: 10.3390/en15176140.

[12] A. Singh, S. C. Satapathy, S. A. Roy, and A. Gutub, "AI-based mobile edge computing for IoT: Applications, challenges, and future scope," *Arab. J. Sci. Eng.*, vol. 47, pp. 9801–9831, 2022. doi: 10.1007/s13369-021-06348-2.

[13] H. Yar, A. S. Imran, Z. A. Khan, and M. Sajjad, "Towards smart home automation using IoT-enabled edge-computing paradigm," *Sens.*, vol. 21, no. 14, pp. 4932, 2021. doi: 10.3390/s21144932.

[14] O. Debauche, S. Mahmoudi, and Z. Guttadauria, "A new edge computing architecture for IoT and multimedia data management," *Inf.*, vol. 13, no. 2, pp. 89, 2021. doi: 10.3390/info13020089.

[15] X. Li, "A cooperative resource allocation model for IoT applications in mobile edge computing," *Comput. Commun.*, vol. 173, pp. 183–191, 2021. doi: 10.1016/j.comcom.2021.04.005.

[16] M. Seifelnasr, R. A. Tawy, and A. Youssef, "Efficient inter-cloud authentication and micropayment protocol for IoT edge computing," *IEEE Trans. Netw. Serv. Manage.*, vol. 18, no. 4, pp. 4420–4433, 2021. doi: 10.1109/TNSM.2021.3103509.

[17] C. Savaglio and G. Fortino, "A simulation-driven methodology for IoT data mining based on edge computing," *ACM Trans. Internet Technol.*, vol. 21, no. 2, pp. 1–22, 2021. doi: 10.1145/3402444.

[18] M. Rohith and A. Sunil, "Comparative analysis of edge computing and edge devices: Key technology in IoT and computer vision applications," in *2021 Int. Conf. Recent Trends Electron., Inf., Commun. Technol. (RTEICT)*, Bangalore, India, 2022, pp. 722–727. doi: 10.1109/RTEICT52294.2021.9573996.

[19] C. Zhan, H. Hu, Z. Liu, Z. Wang, and S. Ma, "Multi-UAV-enabled mobile-edge computing for time-constrained IoT applications," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15553–15567, 2021.

[20] H. Tan, "An efficient IoT group association and data sharing mechanism in edge computing paradigm," *Cyber Secur. Appl.*, vol. 1, pp. 100003, 2021. doi: 10.1016/j.csa.2022.100003.

[21] M. S. Munir, I. S. Bajwa, A. Ashraf, W. Anwar, and R. Rashid, "Intelligent and smart irrigation system using edge computing and IoT," *Complex.*, pp. 1–16, 2021. doi: 10.1155/2021/6691571.

[22] D. Borsatti, G. Davoli, W. Cerroni, and C. Raffaelli, "Enabling industrial IoT as a service with multiaccess edge computing," *IEEE Commun. Mag.*, vol. 59, no. 8, pp. 21–27, 2021. doi: 10.1109/MCOM.001.2100006.

[23] R. Rajavel, S. K. Ravichandran, and K. Harimoorthy, "IoT-based smart healthcare video surveillance system using edge computing," *J. Amb. Intel. Hum. Comp.*, vol. 13, pp. 3195–3207, 2022. doi: 10.1109/MysuruCon55714.2022.9972370.

[24] Z. Song, Y. Hao, Y. Liu, and X. Sun, "Energy-efficient multiaccess edge computing for terrestrial-satellite Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14202–14218, 2021. doi: 10.1109/JIOT.2021.3068141.

[25] D. J. S. Raj, "Optimized mobile edge computing framework for IoT based medical sensor network nodes," *J. Ubiquitous Comput. Commun. Technol.*, vol. 3, no. 1, pp. 33–42, 2021. doi: 10.36548/jucct.2021.1.004.

[26] J. Hwang, "IoT service slicing and task offloading for edge computing," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11526–11547, 2021. doi: 10.1109/JIOT.2021.3052498.

[27] F. A. Doghman, N. Moustafa, I. Khalil, Z. Tari, A. Zomaya and A. Y. Zomaya, "AI-enabled secure microservices in edge computing: Opportunities and challenges," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1485–1504, 2022. doi: 10.1109/TSC.2022.3155447.

[28] H. Hua, Y. Li, T. Wang, N. Dong, W. Li and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023. doi: 10.1145/3555802.

[29] Z. Lv, "AI-enabled IoT-edge data analytics for connected living," *ACM Trans. Internet Technol.*, vol. 21, no. 4, pp. 1–20, 2021. doi: 10.1145/3421510.

[30] X. Chen, M. Li, H. Zhong, Y. Ma, and C. H. Hsu, "DNNOff: Offloading DNN-based intelligent IoT applications in mobile edge computing," *IEEE Trans. Ind. Inform.*, vol. 18, no. 4, pp. 2820–2829, 2023. doi: 10.1109/TII.2021.3075464.

[31] M. Liyanage, P. orambage, A. Y. Ding, and A. Kalla, "Driving forces for multiaccess edge computing (MEC) IoT integration in 5," *ICT Express*, vol. 7, no. 2, pp. 127–137, 2021. doi: 10.1016/j.icte.2021.05.007.

[32] T. Kim, J. Kwak, and J. P. Choi, "Satellite edge computing architecture and network slice scheduling for IoT support," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14938–14951, 2021. doi: 10.1109/JIOT.2021.3132171.

[33] L. Zhang, Y. Zou, W. Wang, Z. Jin, Y. Su and H. Chen, "Resource allocation and trust computing for blockchain-enabled edge computing system," *Comput. Secur.*, vol. 105, pp. 102249, 2021. doi: 10.1016/j.cose.2021.102249.

[34] F. Saeik *et al.,* "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Comput. Netw.*, vol. 195, pp. 108177, 2021. doi: 10.1016/j.comnet.2021.108177.

[35] A. I. S. Stinean, R. E. Precup, E. M. Petriu, R. C. Roman, E. L. Hedrea and C. A. B. Drago, "Extended kalman filter and takagi-sugeno fuzzy observer for a strip winding system," *Expert Syst. Appl.*, vol. 208, pp. 118215, 2022. doi: 10.1016/j.eswa.2022.118215.

[36] S. K. Malchi, S. Kallam, F. A. Turjman, and R. Patan, "A trust-based fuzzy neural network for smart data fusion in Internet of Things," *Comput. Electr. Eng.*, vol. 89, pp. 10690, 2021. doi: 10.1016/j.compeleceng.2020.106901.

[37] S. Ghosh, "Neuro-fuzzy-based IoT assisted power monitoring system for smart grid," *IEEE Access*, vol. 9, pp. 168587–168599, 2021. doi: 10.1109/ACCESS.2021.3137812.

[38] O. J. Pandey, T. Yuvaraj, J. K. Paul, H. H. Nguyen, and K. Gundepudi, "Improving energy efficiency and QoS of lpwans for IoT using q-learning based data routing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 365–379, 2021. doi: 10.1109/TCCN.2021.3114147.