# Smart IoT Framework for Sustainable Cities: Integrating Information Systems and Artificial Intelligence

Amarildo Rista, Alma Stana
Faculty of Information Technology, Aleksander Moisiu University of Durres, Albania

***Abstract*—IoT architecture is an intelligent green city when it incorporates information systems and Artificial Intelligence (AI) to maximize the utilization of resources, upgrade public services, and produce cities very efficient but also environment-friendly. There remain poor feature selection and extraction, unbalanced load, Quality of Service (QoS) issues, and energy consumption by blockchain which prevents effective implementation. This research solves these issues through a multi-step approach. First, we build a smart IoT network that gets the IoT device data from our CICIDS2017 dataset, preprocessed with Normalization to standardize the data. Next, Wrapper-based Feature Selection is applied to identify the most significant features used by the Autoencoder's deeper Feature Extraction to improve model performance. A QoS scheme based on Software Defined Networking (SDN) will dynamically distribute loads with low latency to balance loads and achieve QoS. We further utilize a stateless Q-learning algorithm to avoid congestion in IoT device data distribution. We then use the Hyperledger Fabric for efficient blockchain in combination with Linear Network Coding (LNC) to save energy on the system. To detect cyber-attacks, we adopt a Quasi-Recurrent Neural Network (QRNN) that is Bio-optimized to enhance detection while minimizing false positive responses. Finally, we evaluate the performance of the proposed system on the following metrics: Response Time (ms), Throughput (Mbps), Attack Detection (%), False Positive Rate (%), and Energy Consumption (%). This approach is implemented through NS-3.35 using Python, providing a solid framework for comparative studies and the promotion of sustainable operations of a smart city.**

***Keywords*—Smart city; blockchain; IoT; artificial intelligence; auto encoder; Quality of Service**

## I. Introduction

In the 21st century, accelerated urbanization has imposed tremendous challenges on the sustainable management of urban areas [1]. As more and more cities arise, issues of resource depletion, environmental degradation, and infrastructural inefficiencies become more common [2]. In order to transcend these problems, the convergence of Artificial Intelligence (AI) and the Internet of Things (IoT) within smart city frameworks has been suggested as a solution for improving urban efficiency, resilience, or just the well-being of its citizens [3]. A smart city is considered a city that implements Information and Communication Technologies (ICT) to counter urbanization problems by building, operating, and promoting environmentally sound development practices [4]. It is essentially an intelligent network of devices and appliances that communicate wirelessly and send their data through wireless technologies and cloud computing. This wireless and cloud computing infrastructure enables real-time data gathering and analysis for planners and administrators for informed decision-making to optimize resource utilization and improve public services [5,6,7,8]. The convergence of AI and IoT technologies is being leveraged to resolve the management systems of Smart Cities. These systems provide data-driven answers to urban issues, such as reducing traffic accidents, improving energy efficiency, or detecting maintenance faults of potholes or environmental hazards. A case in point is the study done in Aveiro City, wherein these systems have exhibited their capabilities in offering actionable inputs for improving security, energy efficiency, and sustainability from the perspective of AI and IoT-based solutions for smart cities [9,10]. In addition to doing this, a study has been conducted regarding the integration of blockchain with AI and IoT, with the intention of further development of secure IoT applications. Thanks to this decentralization, essential for data integrity and security, which in turn are fundamental in the credible functioning of smart city infrastructures, blockchain guarantees a reliable environment. Its inclusion drives technological innovation in intelligent and green IoT applications through resilient architecture for urbanization [11,12]. AI and digital technologies represent both opportunities and threats to the construction of smart green units in the Global South. Qualitative research through systematic literature reviews and case studies has addressed the opportunities and challenges issued on digital technologies and AI in this regard. It probed into the policy implications and furnished a framework for smart sustainable cities alongside a call for context-sensitive approaches [13,14]. Recent research has seen frameworks emerge that aim to improve city management and active participation of citizens [15, 16]. The function of AI in intelligent city systems also encompasses other applications such as predictive analytics and infrastructure construction [17, 18]. AI can optimize energy consumption, mitigate traffic congestion, and enhance pollution management through its analysis of huge amounts of data [19]. Although there are potential advantages, the integration of IoT and AI in smart cities also poses risks, most notably data privacy and infrastructure security [20,21]. As cities grow more dependent on AI, managing these risks is important to make sure that technological progress adds value to urban development without undermining citizens' rights or public safety [22]. The research aim is the development of a Smart IoT framework combining

Information Systems and Artificial Intelligence for sustainable smart cities.

### A. Motivation and Objectives

The goal is to improve the efficiency of IoT-based Smart city environments to overcome issues like minimizing congestion, security and privacy challenges, feature selection issues, uneven resource distribution, and so on. Numerous major challenges contemporaneous in existing work are labelled as follows:

- Lack of Proper Feature Selection: Existing research does not incorporate the effective feature selection approaches that lead to minimizing the model accuracy and efficiency.
- Challenges in Feature extraction: In the existing research, the robust feature extraction techniques were absent, which limits the ability to analyse and process the IoT-generated data efficiently.
- Ineffective Load Balancing and QoS Support: Existing research negatively impacts the server load balancing and fails to maintain the essential quality of service (QoS).
- Challenges in Congestion Management: Uneven distribution of IoT resources and congestion in smart city networks remain unresolved, impacting the overall system performance.
- Inadequate Threat Classification and High False Positive Rate: In the existing research, the detection mechanism lacks proper classification models, which leads to an increased false positive rate and minimizes the cybersecurity effectiveness.
- Blockchain challenges: The existing research incorporates the blockchain into IoT-based smart cities, presenting the challenges, especially in terms of high energy consumption and privacy concerns.

Motivated by the above-mentioned issues, we propose to develop a Smart IoT framework combining Information Systems and Artificial Intelligence for sustainable smart cities. This framework aims at resource optimization, feature selection, and extraction, incorporating security, improving data transmission efficiency, and ensuring load balancing and congestion avoidance for sustainable urban development. The enduring purposes of this study are labelled as follows.

- To develop an optimized feature selection approach to enhance model accuracy and efficiency in IoT-based smart city environments.
- Develop a robust feature extraction framework to enhance the data representations and analysis in IoT-based smart cities.
- Design an effective load balancing mechanism to make sure the seamless data routing and enhance the response time while maintaining the QoS.
- Develop an adaptive resource allocation approach to address the uneven distribution and minimize the congestion in IoT-based smart city networks.
- Develop energy-efficient and privacy-preserving blockchain approaches to improve the security and minimize the computational overhead in IoT-based Smart cities.
- Develop an advanced classification model for threat detection with reduced false positive rates, ensuring improved security in smart city infrastructure.

### B. Research Contribution

The primary goal of this research work is the development of a Smart IoT framework combining Information Systems and Artificial Intelligence for sustainable smart cities. The following are the main contributions to this research;

- First, we utilize the CICIDS2017 dataset and then apply the normalization for preprocessing.
- After that, select the features using the wrapper technique, then apply the autoencoder for feature extraction.
- To distribute the load without any delay, we utilize the QoS-based SDN network.
- To avoid congestion when sharing the load to the server, we utilize the stateless Q-learning algorithm.
- After preserving and reducing the energy consumption in blockchain, we utilize the Hyperledger Fabric blockchain model with Linear Network Coding (LNC).
- Finally, we detect the attacks by using the Quasi-Recurrent Neural Network (QRNN) with Butterfly Optimization Algorithm (BOA) in a smart city environment.

### C. Paper Organization

The subsequent components of this task have been arranged as follows. The methodologies and the literature evaluation are presented in Section II. Section III contains the problem statement and description. The proposed research method, along with the protocol and algorithm, is outlined in Section IV. In Section V, simulation results can be obtained alongside a comparison with the proposed method and other methodologies. The suggested method is depicted in Section VI.

## II. Literature Review

This section summarizes and inspects the main research gaps addressed in the existing works. Authors in [20], introduce several approaches of "IoT-fog-cloud for load distribution". Second, effective fog computing techniques for load balancing amongst fog devices are provided. Last but not least, a comparison of the suggested technique's efficiency with that of current methods makes it abundantly evident that it achieves enhanced efficiency, good use of energy, excellent throughput, and efficient use of resources while lowering reaction times. However, as the volume of IoT data continues to rise at an exponential rate, effectively managing, processing, and securing this information will present significant challenges. Authors in [21] suggest a smart, resilient IoT paradigm based on AI. Each of the many services offered in the context of smart cities has begun to play a crucial part in their growth. The service takes

into account non-priority services that serve several areas, as well as route prioritization. Both services will have a highly visible location thanks to the suggested model. Furthermore, in the future, with the advent and proliferation of smart cities, there will be difficulties in achieving robust security; a cutting-edge cryptography-based model will need to be introduced to secure IoT data and infrastructure. Authors in [22] develop a next-generation "IoT-enabled Dynamic Food SCM system for smart cities" that can trace the origins of contaminants in food carrying marketplaces, intelligently route trucks, and guarantee food quality. Additionally, an IoT-based intelligent sensor-based gathering strategy is used, which is intended to increase the supply chain network's functioning effectiveness and precision. This will also increase the dataset's adaptability and achievable size, which will be accompanied by a vehicle the routing technique that tracks the causes of contaminant of contaminated food in the marketplaces. However, managing cloud computational assignments efficiently is going to be a problem. This will require the execution of assignments at the network edge to alleviate delays and conserve energy in smart city settings. Authors in [23] propose an innovative energy control and harvesting method based on hot spot concerns to improve the network lifespan of "IIoT in smart cities". The addition of energy-harvesting nodes in the standard sensor nodes has enhanced the system's "energy-aware cluster" and forwarding procedures. A thorough simulation of the suggested system is conducted under a variety of network topology scenarios. A comparison of the simulated outcomes of several of the techniques will be explored. The substantial improvement in network longevity above the current state has been demonstrated. Furthermore, challenges of load balancing are faced in smart city environments as a result of improper positioning of Energy Harvesting (EH) nodes. Seamless coverage to monitor the target area continuously became a pivotal issue. The number of Cluster Heads (CHs) or EH nodes in the present method requires, therefore, brute-force determination, which decreases scalability and requires intelligent optimization strategies. Authors in [24] suggest a structure for health monitoring and diagnostics for "geo-distributed edge clusters" that handle the large amounts of data produced by apps in smart cities. This system uses edge clusters spread throughout the smart city and is based on the MapReduce methodology for distributed large data processing. Within this architecture, a monitoring tool called SmartMonit is used to gather data on the health of devices at the edge and forecast any malfunctions using a self-organizing map technique based on artificial neural networks. The suggested approach is implemented across various clusters to evaluate its effectiveness in detecting failures. However, in the future, diagnosing failures in resource-constrained IoTs will be challenging due to the complexity of identifying root causes. Hence, the development of fault diagnosis and self-healing schemes based on root because analysis will eventually contribute to ensuring system reliability and resilience. Authors in [25] proposed to identify control over user validity in systems using a dispersed authentication method, which is considered in order to offer a safe framework for the smart computational susceptibility in the long-term sustainability of smart cities. Using a "decentralized authorization algorithm (DAA)", a "secure trust aware philosopher privacy and authentication (STAPPA)" paradigm is illustrated and put into practice. This aids in identifying and countering various smart edge computing threats that compromise data security, privacy, and the authentication of users. Additionally, during the learning phase, a "Genetic Algorithm-based Reinforcement Learning (GARL)" approach is used for optimization of networks, where it looks for abnormalities and finds the shortest path on the network site. Nevertheless, big data in smart cities is facing ballooning security threats from different kinds of attacks and vulnerabilities embedded within fog and cloud computing environments. One of the major issues at hand is to ensure the security, immutability, and decentralization of big data, thus requiring a range of lightweight authentication algorithms that can assist in the detection and mitigation of threats. Authors in [26], present a variety of approaches to create a novel safety plan for systems in smart cities. The produced "hash function, private key, public key, and session key" are all part of the "Collaborative Mutual Authentication (CMA)" method, which is utilized here to verify user identification. Furthermore, to identify network threats and guarantee the privacy of the smart city, a "meta-heuristic genetic algorithm" called Random Forest is employed. In the longer view, adoption of the suggested distributed ledger blockchain framework will be challenging for real-life applications, because it requires enormous funding and resource allocation.

## III. Problem Statement

Specific existing issues, authors in [27], investigated how deep learning-based contemporary agriculture is supported by agricultural information systems in terms of the advantages of agricultural output. The agricultural IoT technology architecture served as the foundation for the suggested information service system. A deep belief neural network-based model for predicting the price of agricultural products was developed. Simulations were used to determine the DBN's ideal structure. Vegetable price prediction in a specific city was used to assess the effects of agricultural technology on the advantages of vegetable cultivation. It was found that the DBN-based agricultural product price projection system was able to estimate the price of vegetables in a given city with a high degree of accuracy. The main issues are given below:

- Their research lacks proper feature selection, which impacts the accuracy and effectiveness of the model.

Authors in [28] proposed to develop smart cities toward a more functional sharing and interconnecting, this study attempts to analyze the IoT and cloud computing-based smart city data technologies. First, IoT data pertaining to smart cities is gathered. Next, the data so gathered is normalized. For machine learning, the IoT paves the way for interaction between devices without human intervention. So, they use a machine learning algorithm known as the Adaptive Random Forest. A multi-criteria optimization is a method that they suggest for effective resource allocation of the virtual machine in Cloud Computing. They analyze and further simulate the use of IoT and Cloud Computing technologies in the development of smart cities to highlight their efficacy. The major issues are given below:

- Their research lacks feature extraction in the IoT-based Smart city environment.

Authors in [29] proposed to improve communication efficiency and dependability with the least amount of delay in terms of energy use and data privacy for data transfer. This study presents "Intelligent Bison-based secure edge-enabled computing IB-SEC)" architecture for smart cities leveraging the IoT. To increase the efficiency, security, and dependability of data transfer in an Internet of Things smart city network, the created IB-SEC platform combines a distributed hashing-based security method with the "African Buffalo Optimization" method. To be able to provide a secure wireless connection and safeguard data from unwanted access, this platform integrates robust edge computing and "MAC (median access control)" protocols for identification, authentication, encryption, and access control. In summary, the IB-SEC framework offers a cutting-edge strategy for safe Internet of Things networks and allows smart city applications. Additionally, the platform that has been designed enables the integration of diverse "edge devices, sensors, and systems" that are efficiently managed and adjusted by MAC protocols. The main issues are given below:

- Their research negatively impacts the load balancing on the server and does not support the quality of services such as data transmission and response time.

Authors in [30] suggested the behaviour of SUs in the context of poor sensing is mathematically modelled. In their suggested DNN-based CR-IoT designs, they presented a complete spectrum preference methodology based on DNN. For an intelligent choice, they have also suggested a novel DNN-based complete spectrum decision method to meet the spectrum decision framework. This allows one to choose, depending on the quality of the accessible channel, whether to proceed with the data communication or to forego the chance to and wait for the subsequent sensing event. When there is a large density of IoT devices in a relatively tiny area with an elevated need for spectrum, their study will prove extremely helpful. Here, the primary challenges are given below:

- Their research fails to effectively address the challenges in the uneven distribution of resources and congestion in the IoT-based smart city network.

Authors in [31] proposed the OMLIDS-PBIoT approach, a unique optimum "Machine Learning-based intrusion detection system" for secure BioIoT in a smart city setting, which has been created in this paper. Data pre-processing is carried out in the subsequent step to convert data into the proper format, and the OMLIDS-PBIoT approach functions by employing BC and ML techniques. Additionally, an FS model that utilizes "golden eagle optimization (GEO)" is created in order to extract relevant subsets. For intrusion categorization, the "random vector functional-link network (RVFL)" and HBO models were used. In turn, the use of blockchain is used for safe data transfer in the context of IoT-enabled smart cities. Here, the primary challenges are given below:

- Their research lacks proper classification of threat detection and fails to minimize the false positive rate.
- Their research faces challenges in blockchain implementation, especially concerning energy consumption and privacy issues.

Recent studies have further expanded the scope of AI-enabled sustainable urban systems. Faisal et al. [32] integrated machine learning with multi-criteria decision making to assess AI-driven personalization in smart cities. Huy and Phuc [33] examined how technologically vigilant leadership supports smart, sustainable circular supply chain management in Industry 6.0 environments. Xiao and Dong [34] optimized AIGC technology for IoT devices using deep learning, highlighting efficient AI deployment on resource-constrained urban edge nodes.

### A. Research Solution

The Wrapper technique is used for feature selection and autoencoder for feature extraction in IoT-based smart city environments. A QoS architecture based on SDN balances server load distribution and manages response time and data transmission. Stateless Q-learning avoids congestion. Hyperledger Fabric blockchain preserves privacy and reduces energy consumption. Quasi-Recurrent Neural Network and butterfly optimization algorithm are combined for attack classification and accuracy in IoT-based smart city environments.

## IV. Proposed Work

The primary goal of this research work is the development of a Smart IoT framework combining Information Systems and Artificial Intelligence for sustainable smart cities. Fig. 1 represents the overall architecture of the proposed methodology. As part of this strategy, the following key steps are taken

- Information Gathering using IoT devices
- Feature Selection and Feature Extraction
- Quality of service-based load distribution
- Congestion avoidance in a smart city environment
- Privacy-preserving, energy consumption-based smart city using blockchain
- Artificial Intelligence-based attack detection in smart cities

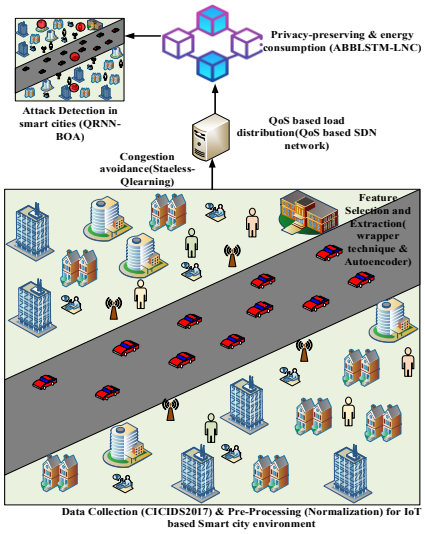


Fig. 1. The overall architecture of the proposed methodology.

The end-to-end workflow of the proposed framework operates in two coordinated evaluation paths. In the intrusion-detection path, CICIDS2017 data are normalized, filtered through wrapper-based feature selection, encoded by the autoencoder, and classified by the BOA-optimized QRNN to detect cyber-attacks. In the network-simulation path, NS-3.35 generates smart-city IoT traffic across SDN-enabled servers, where the controller performs QoS-based load balancing and stateless Q-learning reduces congestion. Detected security events and validated network transactions are recorded in Hyperledger Fabric, while Linear Network Coding reduces blockchain transmission overhead and energy consumption. This separation ensures that each evaluation component is supported by an appropriate data source and simulation environment.

### A. Information Gathering Using IoT

Initially, we collect the dataset for our framework. We utilize the CICIDS2017 dataset. Although CICIDS2017 is an intrusion-detection benchmark derived from controlled network traffic rather than direct smart-city IoT operational traces, it is used in this study to evaluate the cyber-attack detection component of the proposed framework. Network performance metrics, including response time, throughput, congestion control, and energy consumption, are evaluated separately through NS-3 simulations, as described in Section V. Before further processing, we implement the normalization approach to standardize data values, make sure that variations in scale do not affect the model performance. By employing this dataset with efficient preprocessing, we can design the real-world scenario and improve smart city resilience against potential threats. Normalization

Normalization is a preprocessing step to scale input data so model has ranges consistent (increased performance). A standard formula for normalization is:

$$M_{norm} = \frac{M - M_{min}}{M_{max} - M_{min}} \quad (1)$$

where,

$M$ is the initial feature value,

$M_{min}$ is the minimum value of all of the feature,

$M_{max}$ is the maximum value of the feature, and

$M_{norm}$ is the normalized feature value.

This normalizes all feature values between 0 and 1 ensuring that higher magnitude features.

### B. Feature Selection and Feature Extraction

With an extensive dataset, the next crucial step is identifying the most significant features. So, we utilize the Wrapper based feature selection approach, we filter out redundant and irrelevant data, ensuring that only the most important attributes are utilized for analysis. To further refine the process, we apply the Autoencoder for feature extraction, enhancing the model performance and achieving the high accuracy. This step not only improves the predictive capabilities but also enhance the overall effectiveness in decision-making within the smart city environment.

*1) Wrapper-based feature selection:* Wrapper-based feature selection is an iterative algorithm that analyzes sets of features by training a machine learning model and then assessing its accuracy. The method reconciles accuracy with computational power by picking the most relevant features and eliminating redundant and irrelevant information. The CORRAccuracy algorithm is an example of a highly acclaimed wrapper-based approach that uses correlation-based filtering and accuracy assessment to pick optimal features.

The CORRAccuracy measure estimates the quality of features selected based on the following formula:

$$CORR = \frac{V \times avg(corr_{hc})}{\sqrt{V + V(V-1) \times avg(corr_{hh})}} \quad (2)$$

where,

$corr_{hc}$ is the feature-class correlation,

$corr_{hh}$ is the correlation between features, and

$V$ is the number of features selected.

This measure ensures that the chosen features have high correlation with the target class while reducing inter-feature redundancy. After correlation is calculated, the accuracy measure (*accuracy*) also confirms feature importance:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

Here,

$TP$ denoted as true positive

$TN$ denoted as true negative

$FP$ is denoted as false positive

$FN$ is denoted as false negative

The wrapper method is one where training and evaluation are repeated cycles in order to locate the optimal subset of features that maximizes classification performance. Yet another critical feature selection equation is the information gain ($IG$), which measures the extent to which a feature reduces uncertainty:

$$IG = I(A) - I(A|B) \quad (4)$$

where,

- $I(A)$ Represents the target variable of entropy
- $I(A|B)$ Denotes the entropy of the target given the feature $B$

Mutual information ($MI$) is also used for ranking features by quantifying the dependence between a feature and the class:

$$MI(A,B) = \sum_a \sum_b JP(a,b) \log\left(\frac{JP(a,b)}{JP(a)JP(b)}\right) \quad (5)$$

Here,

- $MI(A,B)$: Mutual Information between feature A and target class B.
- $\sum_a \sum_b$: Summation over all possible values of a and b.

- *JP*(*a*,*b*): Joint probability distribution of feature A and class B (i.e., the probability that feature A takes value a and class B takes value b at the same time).
- *JP*(*a*): Marginal probability of feature A (i.e., the probability that A=a happens).
- *JP*(*b*): Marginal probability of class B (i.e., the probability that B=b happens).
- *a*: Denotes an instance of the feature A.
- *b*: Denotes an instance of the class B.

In order to optimize the feature subset, the feature ranking function combines correlation and accuracy as:

$$\text{Rank}(l) = \beta \times \text{CORR}(l, \text{class}) + \emptyset \times \text{Accuracy}(l) \quad (6)$$

- *Rank(l)*: Final ranking score of feature *l*.
- *l*: Individual feature being examined.
- $\beta$: Correlation metric weighting factor (governs the relative importance of correlation in ranking).
- *CORR*(*l*, *class*): Feature *l* correlation with the target class.
- $\emptyset$: Accuracy metric weighting factor (governs the relative importance of accuracy in ranking).
- *Accuracy(l)*: Model accuracy when the feature *l* is used in the classifier.

This formula orders features by blending correlation with the target class and with the classification accuracy obtained using the feature. The weights $\beta$ and $\emptyset$ enable the fine-tuning of the impact of each measure in the ordering. For $\beta > \emptyset$ correlation is given greater precedence, and $\emptyset < \beta$ gives a greater precedence to accuracy. This strikes a balance between choosing features that are informative as well as beneficial towards model performance.

*2) Autoencoder for feature extraction:* A unique kind of multilayer perceptron called an autoencoder has the same number of neurons in both its input as well as its output layers. Two elements of the autoencoder's design, the encoder and the decoder, are taught at every succeeding layer. In a deep autoencoder, each layer receives input from the layer before it. Fig. 2 represents the autoencoder architecture. In particular, the autoencoder has been taught to convert the raw input into a compressed representation, from which the output is rebuilt.

$$I(y) = \varphi(\mathbb{C} \cdot z(y) + c) \quad (7)$$

where,

$I(y)$ is the encoded of the input $y$, $z(y)$ is the input data, $\varphi$ is the activation function, the acquisition function in this instance is sigmoid. The letters "weight" and "bias" are $\mathbb{C}$ and $c$, correspondingly.

$$S(y) = b(\mathbb{C} \cdot \text{H}(y) + c') \quad (8)$$

H($y$) is denotes latent representation produced by the encoder.

$c'$ is denoted as bias implemented during decoding.

$b$(.) is used to ensure the output stays within a range.

$S(y)$ is the predicted output of the input $I(y)$ based on the latent representation H($y$). The weights of the autoencoders are optimized to reduce the inaccurate reconstruction of the network. The reconstructed error is calculated as follows:

$$(z(y), S(y)) = \|\text{E}(y) - S(y)\|^2 \quad (9)$$

Here, E($y$) is the actual input data. $\sum_{E=1}$. It represents the summation and the index starts with first element. Furthermore, the cross-entropy measure's reconstruction error for binary values (7):

$$\text{h}(z(y), S(y)) = -\sum_{E=1} \left[ z(y)_E \log S(y)_E + \left(1/(y)_E\right) \log\left(1 - S(y)_{s_E}\right) \right] \quad (10)$$

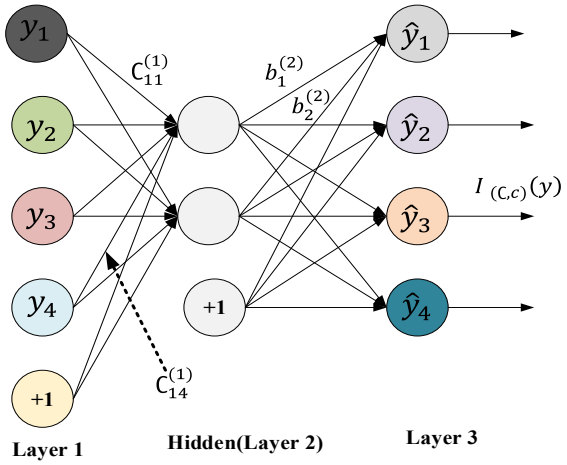


Fig. 2. Autoencoder.

## C. Quality of Service-Based Load Distribution

Based on the selected and extracted features, the system processes only the most relevant data, minimizing computational overhead and enhancing the response times. However, smart city generates an enormous volume of data, leading a probable server congestion and latency. To address this, we employ the Quality of Service (QoS) mechanism using software defined networking (SDN). This technique ensures the dynamic load balancing, optimizing server response time and effective data transmission. By distributing the loads, our framework mitigates the bottlenecks, enabling real-time decision making across the city's interconnected system.

SDN was selected over conventional distributed routing and static load-balancing approaches because it provides centralized visibility of network state, policy-based QoS enforcement, and programmable traffic engineering in large-scale IoT deployments. Compared with traditional IP-based management, SDN enables faster adaptation to congestion, more efficient resource allocation across servers, and lower end-to-end latency under dynamic smart-city traffic conditions.

Quality of Service (QoS) mechanism using software-defined networking (SDN).

In the Software Defined Networking (SDN) based Quality of Service (QoS) mechanism for load balancing, the main objective is to dynamically allocate incoming requests to multiple servers so that server response times are optimized and data transmission is efficient. Since smart city systems produce a vast amount of data, the SDN architecture is responsible for preventing server overload and reducing latency. In order to realize dynamic load balancing, the SDN controller is always observing the workload of every available server and sends the incoming requests to the server that has the lightest workload. The following describes the load distribution with a mathematical equation:

$$R_o = \min(R_1, R_2, \ldots, R_p) \tag{11}$$

where,

$R_o$ is the load for the server selected and $R_1, R_2, \ldots, R_p$ describes the present loads on all the available servers. The formula keeps, in a dynamic manner, forwarding new requests to the server with the least load and thereby avoids overloading any single server. By constantly evaluating the load, the system developed in SDN efficiently spreads traffic across all available resources, ensuring no bottlenecks are created.

In addition, there is another load threshold condition to further enable optimization of load distribution. Each of the servers has been given a maximum load past which it cannot be considered efficient; when the load on that server exceeds this threshold, requests are diverted to other servers that are less loaded. This is what is mathematically expressed as:

$$R_o \leq R_{max} \tag{12}$$

$R_o$ load the server $o$ while $R_{max}$ is an allowable maximum load. It stops any single server from being saturated or loaded, giving the system the ability to dynamically reroute any excess requests forwarded to it, thus optimizing request processing and availability of continuous service.

A key performance measure in load distribution under QoS is response time $Res_{time}$, defined as the time elapsed from the receipt of a request until a response is issued. It is calculated with the formula:

$$Res_{time} = time_{end} - time_{start} \tag{13}$$

where, $time_{start}$ represents the time at which the request is received, and $time_{end}$ corresponds to the moment when the system finishes processing and returns. Reducing response time is critical in ensuring real-time decision-making capacity, particularly in smart city functions where there is a need for quick processing of activities like traffic monitoring and emergency response. Through the dynamic allocation of requests to idle servers, the SDN controller minimizes delays and maximizes the capacity of the system to respond quickly to arriving data.

Another key system performance metric is server utilization, which measures the percentage of a server's processing power being utilized. This is determined by the formula:

$$U_o = \frac{time_{busy}}{time_{total}} \times 100 \tag{14}$$

where $U_o$ represents the utilization percentage of the server $o$, $time_{busy}$ is the time the server spends actively processing requests, and $time_{total}$ is the overall observation time. The measure helps in verifying the resource optimization of the servers. To distribute the workload among several servers, the SDN controller prevents overloading of any server and keeps most of the running states steady and effective. The method also helps in synonyms scaling the system to accommodate the increasing data load without any performance degradation.

An additional load difference function runs within the SDN controller to maintain the load balance over multiple servers. This can be represented mathematically as:

$$\Delta Y = \left| Y_q - Y_o \right| \tag{15}$$

where, $\Delta Y$ is the difference in load between the server $q$ and server $o$. Whenever that difference exceeds a pre-set limit, the SDN controller dynamically allocates incoming requests from the overwrought server into the less-loaded server place. Such load management in advance avoids congestion, increases server performance, and guarantees that the entire system keeps its responsiveness uniform even when traffic fluctuates over a certain limit.

Further, the system analyzes its performance based on the data rate of transmission, i.e., the rate at which data is transferred successfully within a time limit. It is determined through the formula:

$$D = \frac{S_{transmitted}}{time} \tag{16}$$

The data transmission rate is expressed as $D$ with $S_{transmitted}$ being the number of packets that have been successfully transmitted, and time being the observation time of data transmission. The faster the data transmission rate translates to a larger amount of data processed effectively by the system, thereby enhancing the quality of service for smart city infrastructures. Through constant checking and fine-tuning of the data transmission rate, the SDN controller encourages rapid and timely transmission of data and supports real-time data applications in surveillance, traffic flow, and public safety.

The QoS-based load distribution scheme based on SDN efficiently meets the demands presented by the mass data generation of smart city applications. With the deployment of dynamic load balancing equations, such as the minimum load selection, load threshold condition, response time computation, server utilization tracking, load difference function, and data transmission rate, the system provides enhanced server performance, less latency, and effective resource management. This integrated approach allows the SDN framework to have high responsiveness and reliability levels, supporting real-time decisions and improving the general level of service quality in smart city interconnecting systems.

### *D. Congestion Avoidance in Smart City Environment*

The uneven distribution of resources leads to congestion and unhitches the whole system of smart city operations. To combat this congestion, we will deploy stateless Q-learning, a reinforcement learning-based approach that will dynamically adjust the allocation of computational resources. Thus, this becomes a decentralized technique that will involve building up

a congestion avoidance methodology that will work in the shadows to prevent any conceivable congestion from disrupting IoT services like traffic management, energy distribution, and emergency response from functioning smoothly.

*1) Stateless Q-learning:* Stateless Q-learning is a reinforcement learning technique that can be applied in situations where it is not practical to store a complete representation of the system state. In an IoT smart city scenario, with millions of IoT devices constantly producing computational loads, stateless Q-learning is perfect for real-time congestion avoidance. It enables IoT devices to make autonomous adjustments to their computational resource allocation without needing a central controller, lessening communication overhead and ensuring real-time response.

The Q-learning here operates by modifying a Q-value—a value that estimates the expected future reward gained by performing an action. For every IoT device $\delta$ at time step, the Q-value will be calculated by the following formula:

$$Q_i^{\delta}(time)=\big(1-X_{\delta}(time)\big)Q_i^{\delta}(\tau\text{-}1)+X_{\delta}(time)l_i^{\delta}(time) \quad if\ i=U_{\delta}(time) \tag{17}$$

Otherwise,

$$Q_i^{\delta}(time)=Q_i^{\delta}(time\text{-}1) \tag{18}$$

Here,

$Q_i^{\delta}(time)$ is the current Q-value for action $(i)$ by device $\delta$ at time.

$X_{\delta}(time)$ is the learning rate, which determines how much of the new information replaces the old Q-value.

$U_{\delta}(time)$ is the action selected by device $\delta$ at time *time.*

$l_i^{\delta}(time)$ is the reward received by device $\delta$ for taking action $i$.

Reward function to optimize for completing tasks in a minimum amount of time. At a device $\delta$, reward at a time is determined by:

$$l_i^{\delta}(time)= TH_{\delta}\text{-}T_{\delta}(time) \tag{19}$$

Here,

$TH_{\delta}$ is the threshold of time by which the task has to be achieved.

$T_{\delta}(time)$ is the real-time taken by the device to get the task accomplished.

A positive reward is issued if the task is achieved within the threshold time, and a negative reward is issued if the task takes more time than the designated time. It encourages the system to choose such actions that bring down congestion as well as lessen latency.

The agent chooses actions based on an $\varepsilon$-greedy policy to trade off exploration and exploitation. With probability $1\text{-}E_{time}$, the best action according to Q-value is chosen, and with probability $E_{time}$, a random action is sampled:

$$U_{\delta}(time)=\begin{cases}\arg\max Q_i^{\delta}(time\text{-}1) \ with\ probability\ \ 1\text{-}E_{time}\\ random\ action\ \ with\ \ probability\ E_{time}\end{cases} \tag{20}$$

where,

$$E_{time}= E_0 \times time^{-1/\delta} \tag{21}$$

Here, $E_{time}$ reduces as the algorithm continues, enabling early exploration and subsequent exploitation of the best-known actions.

The learning rate $X_{\delta}(time)$ is given as:

$$X_{\delta}(time)= \left[\mu+ \Delta_{\tau}^{\delta}\big(U_{\delta}(time)\big)\right]^{-\omega} \tag{22}$$

where,

$\mu$ is a constant regulating the initial learning rate.

$\Delta_{time}^{\delta}$ is the number of times action

$U_{\delta}(time)$ has been chosen up to time *time.*

$\omega$ is a parameter regulating the rate at which learning diminishes.

As the learning rate decays over time, the policy gradually stabilizes and selects actions that reduce congestion under the current network conditions. Decentralized decision-making is facilitated by this that IoT devices optimize resource usage autonomously, minimizing congestion and maximizing the system's efficiency overall. With the application of stateless Q-learning, intelligent city infrastructure can dynamically distribute computation resources on a network of IoT devices to provide low-latency task execution and prevent service failure in high-priority domains such as traffic control and emergency response. Fig. 3 represents the Q-learning.

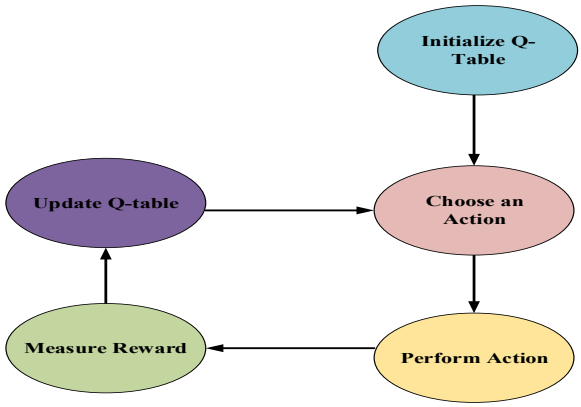


Fig. 3. Q learning.

### *E. Privacy Preserving-Energy Consumption-Based Smart City Using Blockchain*

Blockchain technology promotes heightened security in smart city ecosystems, and it also introduces added challenges pertaining to energy consumption and privacy preservation. Such problems can be addressed by merging the Hyperledger Fabric blockchain with Linear Network Coding (LNC). HLFB ensures robust privacy protection of transactional data analysis and security, while Linear Network Coding improves on energy-

efficient transmission. The combination of both enhances security without compromising sustainable blockchain smart cities.

*1) Hyperledger fabric blockchain:* In the Hyperledger Fabric blockchain system, an efficient privacy-preserving mechanism is created through several cryptographic methods and operational stages. In the initialization stage, key generation and authentication security is taken care of by the Membership Service Provider (MSP) using random numbers and a hash function. The process of registration entails creating and authenticating unique passwords and identifiers through clients, smart contracts, and endorsing peers. For example, in the process of smart contract registration, a verification message $vf_1$ is generated using the following equation:

$$vf_1 = \left(e\left(r_x^* F\right) \| F\right) \oplus \mathrm{h} \tag{23}$$

Here, $r_x^* F$ represents the hashed smart contract ID; $F$ is any random value, while h is the verification parameter. Now, this message will be kept by the MSP and committing peers for future verification.

A similar verification message $vf_2$ is generated at the time of client registration:

$$vf_2 = \left(e\left(Ct_x^*\right) \| rv\right) \oplus avf \tag{24}$$

where, $Ct_x^*$ the client's is hashed ID, $rv$ is a random value, and $avf$ is the related verification parameter. This maintains strict verification procedures for each entity that registers to prevent any unauthorized access. Encryption of data protection is carried out using the Advanced Encryption Standard (AES) technique, which has a key size of 128 bits, where $E_i$ would be the encrypted information as follows:

$$E_i = E(rd, ek) \tag{25}$$

where,

$rd$ is record while $ek$ is encrypted key. For maintaining better privacy, the key $ek$ itself is generated using an interpolation-based approach:

$$ek = YI \oplus (C * F) \tag{26}$$

Here, $YI$ is an interpolated value that is computed as:

$$YI = YI_1 + \frac{(YI_2 - YI_1)}{(XI_2 - XI_1)} * (XI - XI_1) \tag{27}$$

$XI$ refers to the input value that you are finding the interpolated output for.

$XI$ : The lower boundary (beginning point on the X-axis).

$XI_1$: The upper boundary (terminal point on the X-axis).

$C$ is computed as

$$C = 60X^3 + 6X^2 + 6 \tag{28}$$

This way advanced calculations ascertain that the encryption keys are cryptographically derived with the uniqueness of being different from one-another and the keys cannot be accessed by third-party entities externally. Also, the authentication process involves generating and validating digital signatures:

$$sg = (e(mn_x F) \| j) \bmod k \tag{29}$$

$sg$ is the digital signature.

$mn_x F$ is message data augmented with a random value or system-defined parameter

$e(\cdot)$ is the application of a hashing or encoding function to secure the input.

$\|\ \|$ is concatenation of the encoded message with some other value

$j$ (which can be a random nonce or session-specific identifier).

$mod\ k$ makes sure that the signature is computed in a modular arithmetic framework, which is usually employed to restrict the length of the signature and increase cryptographic security.

This guarantees the integrity of the participants in a blend of hashing and modular arithmetic. The authenticity of every message of authentication is guaranteed by means of time-sensitive verification, providing yet another security layer to the process.

In total, the Hyperledger Fabric blockchain platform employs these strict mathematical operations to provide safe registration, data encryption, and verification, facilitating strong privacy protection in smart city systems.

*2) Linear network coding:* In linear coding, the source node divides the data into several blocks. There are $n$ packets, referred to as native packets, in each block. The symbol for this is $z_k$, Z{1,2,…,$n$}. As a result, the source node sends the LNC-coded packet $z_j^{'}$ to the subsequent CH. The mathematical representation of this packet $z_j^{'}$ is provided by Eq. (30).

$$z_j^{'} = \sum_{k=1}^{n} D_{jk} y_k \tag{30}$$

where,

$D_{jk}$ is matrix of coefficients

$z_k$ represent $k$ th native packet.

Z{1,2,…,$n$} is denotes that Z belongs to the set {1,2,…,$n$}

$n$ denotes as total number of packets in the block.

A Galois Field is used to perform operations such addition as well as multiplication. Code vector, $\vec{D}_j = \left(D_{j1}, D_{j2}, \ldots, D_{jn}\right)$ and the block ID are contained in the data packet's $z_j^{'}$ header. The starting value of $n^{'}$ is stored in a counter, and its value is bigger than $n$. The source node keeps track of this counter. The counter is reduced by one unit for every coded packet that is passed on to the next node. The random distribution of the coded data packets goes on until the value of the counter reaches up to Zero. The recruiting and forwarding operations are performed at every single hop. The destination CH initiates the formation of the next cluster upon receiving the packets of the same block. Now, this receiving cluster is converted into a new transmitting cluster that

contains the same nodes. The transmitting CH first schedules the corresponding time at which the node should transmit the encoded data to the receiver cluster, and it is utilized by each node of the transmitting cluster. The receiving CH then captures control of the surrounding nodes to constitute the receiver cluster, and to select the nodes with the higher cost $D_j$. This $D_j$ variable is specified for a node $j$, which may possibly be used for the receiver cluster, as described in Eq. (31).

$$D_j = \sum I_{jk}\left(1-Q_{jk}\right) \forall \text{ node } j \varepsilon \text{ sending cluster} \quad (31)$$

where,

$Q_{jk}$ – Probability of loss

$I_{jk}$ - The indication's function is to show if the sensor is available to receive the packets.

Since it is reserved to receive or transmit data packets for another cluster from a different route, these may not be available. Every neighbor's ping sample is used to periodically test the probability $Q_{jk}$. In order to send the data, $I_{jk}$ depends on the CH transmission method, which designates the sensor nodes. As a result, the transmitting cluster selects the sensors that are anticipated to receive more data as a receiver cluster.

Assume for the moment that a node has acquired the encoded data $z$. The newly encoded data packet can then be retrieved using Eq. (32) as well as Eq. (33).

$$z'' = \sum_{j=1}^{n} D_j z_j' \quad (32)$$

where,

$D_j$ is arbitrarily generated numbers

$z''$ is the native data packets

$$z'' = \sum_{j=1}^{n} D_j \left(\sum_{j=1}^{n} D_{jk} y_k\right) = \sum_{k=1}^{n} \left(\sum_{j=1}^{n} D_j D_{jk} y_k\right) z_j = \sum_{k=1}^{n} g_k z_k \quad (33)$$

When sending the encoded data packet, a sensor will include a new code vector, represented as $\vec{g} = (g1, g2, \ldots, g_n)$, within the packet's $z''$ header, much like the transmitting node does. The system will first determine if the data is new when it receives encoded data. Only when a data set is linearly independent of previously received data from the same block that the sink node has received is it deemed new and fresh. The data will be immediately deleted if it is not new. Thus, the receiver node can retrieve transformative data packets as long as fresh packets are collected. Decoding at the sink node entails applying the Gaussian technique to solve the group of linear equations. If the rank of the matrix is known, the linear equation specifies the solution technique.

$$\begin{pmatrix} D_{11} D_{21} \cdots D1_n \\ D_{21} D_{22} \cdots D2_n \\ \cdots \cdots \cdots \cdots \\ D_{n1} D_{n2} \cdots D_{nn} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \cdots \\ z_n \end{pmatrix} = \begin{pmatrix} z_1' \\ z_2' \\ \cdots \\ z_n' \end{pmatrix} \quad (34)$$

where,

$z_1'$ is an encrypted packet

$(D_{k1}, D_{k2}, \ldots, D_{kn})$ is the code vector

$z_k$ denotes native packets

The sensors in the sending cluster may have received two or more packets from the previous cluster, allowing them to create their own combination. Using randomly selected coefficients, the sensor nodes combine all received packets from the same block to produce a random linear combination.

### F. Artificial Intelligence-Based Attack Detection in Smart Cities

Cyber security is among the major priorities of all smart city strategies. In order to enable effective detection and classification of cyber threats with minimal false positives and maximization of attack detection accuracy, we apply a Quasi-Recurrent Neural Network (QRNN). To boost performance, the Butterfly Optimization Algorithm (BOA) is integrated, which results in an accelerated learning process and higher classification speed. Thus, the synergy between QRNN and BOA provides an extremely responsive and intelligent defense mechanism against new cyber threats for smart city networks.

*1) Quasi-Recurrent Neural Network (QRNN):* The Quasi-Recurrent Neural Network (QRNN) is a hybrid deep model that exploits the fast feature extraction of convolutional layers and the sequential modeling capacity of recurrent networks. The architecture lends itself especially well to smart city application areas where attack detection must be computationally efficient and precise. By providing speed in computation while largely decoupling the computations across time steps, QRNN achieves enhanced performance without leaving the ability to handle sequential data.

QRNN was preferred over conventional CNN, LSTM, and Random Forest classifiers because it offers a better balance among detection accuracy, inference latency, and deployment cost in high-volume IoT traffic. Compared with LSTM-based models, QRNN reduces sequential computation overhead while preserving temporal dependency modeling. Compared with standard CNN and tree-based methods, it achieves stronger attack classification performance on sequential network-flow data with lower false positive rates, making it more suitable for real-time smart-city security monitoring.

The architecture consists of three main components: a convolutional layer that extracts features, a gating mechanism that regulates the flow of information, and a recurrent pooling layer that serves to maintain sequential dependencies

QRNN computes the victim of an attack against a data set, which entails feeding in an input sequence $P = [P_1, P_2, \ldots, P_{time}]$ where each $P_t$ is an input feature at the time step. The convolutional layer convolves many filters over this sequence to derive spatial features. The output of the convolutional layer is given by:

$$H_t = \sigma\left(k * P_{time} + c\right) \quad (35)$$

Here, $H_{time}$ is the feature extracted at time, $k$ is a convolutional filter, $c$ is the bias and $*$ refers to the operation of convolution. The activation function $\sigma$ (usually sigmoid) provides the non-linearity and assists with detecting complex

patterns. The mechanism of the gate also improves upon information flow by deciding which feature should be given to the following layer. The gate value is calculated as:

$$v_{time}=\sigma\ (k_v * P_{time}+ c_v) \tag{36}$$

Here, $v_{time}$ represents the gate value, $k_v$ is the filter for the gating function, and $b_v$ is the bias. This mechanism serves to control the information passed from time steps, keeping out unnecessary or noisy data from passing through the network. The recurrent pooling operation aggregates the current input and the previous hidden state, enabling QRNN to keep memory along sequences. This is expressed by the following equation:

$$g_{time}= F_{time}\odot C_{time-1}+ (F_{time})\odot H_{time} \tag{37}$$

In this equation, $C_{time}$ represents the current memory cell at time, $F_{time}$ is the forget gate that determines the amount of previous memory $C_{time-1}$ that is preserved, and $H_{time}$ is the output of new features. The notation $\odot$ is used for element-wise multiplication. Finally, the results of the LSTM unit are also acquired through a gate controlling the visibility of their internal memory to layers outside:

$$V_{time}=O_{time}\odot C_{time} \tag{38}$$

where, $V_{time}$ is the LSTM output, $O_{time}$ being the output gate, which decides to what extent internal memory is exposed: it is the output that is passed on for doing attack detection, where the LSTM decides whether an incoming data stream is that of normal operation or a potential instance of cyber-attack.

*2) Butterfly Optimization Algorithm (BOA):* In the improvement of the efficiency of the QRNN, Butterfly Optimization Algorithm (BOA) is employed for optimizing its hyper parameters. These habits by which the butterflies search for food and mate are really nature-inspired, coupled with movements driven by a sensory modality known as fragrance. The algorithm maintains a balance between global exploration and local exploitation in order to enhance the accuracy of the model as well as avoid false positives. The fragrance that signifies the objective function or the fitness is calculated as follows:

$$K= cI^A \tag{39}$$

In this equation, $K$ is the sensed fragrance, $c$ is the modality of the senses (a constant that determines sensitivity), I is the intensity of the stimulus, and $A$ is the power exponent that determines the rate of absorption. The flight of a butterfly during the global search phase, when it moves towards the optimum in the population, is described by:

$$\zeta_i^{\varsigma+1}=\zeta_i^{\varsigma}+ \left(\psi^2\times J^*-\zeta_i^{\varsigma}\right)\times K_i \tag{40}$$

Here, $\zeta_i^{\varsigma+1}$ is the updated position of the $i$th butterfly in iteration $\varsigma+1$, $J^*$ represents the best solution found so far, and $\psi$ is a random number between 0 and 1 to ensure stochastic search. This equation facilitates global searching of the best hyperparameters by modifying the weights and biases of the QRNN. For the local search step, in which the algorithm searches in the vicinity of solutions, the position of the butterfly is updated with:

$$\zeta_i^{\varsigma+1}=\zeta_i^{\varsigma}+ \left(\psi^2\times\zeta_j^{\varsigma}-\zeta_k^{\varsigma}\right)\times K_i \tag{41}$$

In this formula, $\zeta_j^{\varsigma}$ and $\zeta_k^{\varsigma}$ are two random butterflies selected from the population. This local search improves the ability of the algorithm to fine-tune the model's performance by searching for close-to-optimal solutions. The algorithm assesses the fitness of every butterfly (or solution) based on the classification error rate:

$$\text{fitness}(\zeta)=\frac{\text{number of misclassified samples}}{\text{Total samples}}\times 100 \tag{42}$$

This task estimates the goodness of a specific solution, in which a lower error rate is reflective of improved model performance. Iteratively, BOA reduces this error, hence achieving increased accuracy and improved generalization to unobserved data.

The combination of QRNN with BOA produces a robust and smart cyber-attack detection system for smart cities. The QRNN processes big-scale sequential data effectively while keeping computational efficiency, and the BOA optimizes the performance of the model by dynamically tuning hyperparameters. This integrated system gives a strong, adaptive defense system against new cyber threats, which keeps smart city networks resilient and secure. Algorithm 1 describes the pseudocode for the butterfly optimization algorithm.

**Algorithm 1: Butterfly Optimization Algorithm (BOA)**

Input: N (population size), Max_iter (max iterations), c (sensitivity), A (absorption rate), QRNN (model), X, Y (data)

Initialize butterfly population (Z) with random QRNN hyper parameters

Evaluate fitness

fitness(ζ) = (Misclassified samples / Total samples) × 100

Set $J_{best}$ = best solution with minimum fitness

For ς=1 to Max_iter:

For each butterfly $\varsigma_i$:

$K= cI^A$ # calculate fragrance

If random(0, 1) < p: $\zeta_i^{\varsigma+1}=\zeta_i^{\varsigma}+ \left(\psi^2\times J^*-\zeta_i^{\varsigma}\right)\times K_i$ # Global search

Else:

Select $\zeta_j^{\varsigma}$ a $\zeta_k^{\varsigma}$ from population,

$\zeta_i^{\varsigma+1}=\zeta_i^{\varsigma}+ \left(\psi^2\times\zeta_j^{\varsigma}-\zeta_k^{\varsigma}\right)\times K_i$ # Local search

Update J_best if a better solution is found

Return J_best (optimized QRNN hyperparameters) and fitness(J_best)

## V. Experimental Results

This section presents the performance evaluation and experimental evaluation of the suggested study strategy. This part is divided into three subsections: research summary, comparative analysis, and simulation study.

### A. Simulation Study

The proposed study approach is replicated using NS 3.35 with Python. This tool works well and offers all the specifications needed for the recommended approach. In Tables I and II, the system specification and parameters are presented.

TABLE I. SYSTEM SPECIFICATIONS

| | | |
|---|---|---|
| Software Specifications | OS | Ubuntu 22.04 |
| | Network Simulator | NS 3.35 with Python |
| Hardware Specifications | RAM | 4GB |
| | Hard Disk | 500GB |

TABLE II. SIMULATION PARAMETER

| Parameters | | Descriptions |
|---|---|---|
| Network Parameters | IoT Devices | 50 |
| | Base station | 2 |
| | Blockchain node | 1 |
| | Server | 3 |

### B. Intrusion Detection Experimental Setup

The QRNN-based attack detection experiments were conducted on the CICIDS2017 dataset using an 80/20 stratified train-test split. After preprocessing and normalization, 500,000 flow records were used for training and evaluation across 14 attack classes. Wrapper-based feature selection retained 45 features, which were encoded by an autoencoder with a 32-dimensional latent representation. The QRNN consisted of two convolutional layers (128 and 64 filters), a quasi-recurrent pooling layer, and a dense output layer. BOA optimized the learning rate (0.0001–0.01), batch size (16–128), and number of epochs (10–50) over 30 iterations with a population size of 20. The final model was trained with a batch size of 64, a learning rate of 0.001, and 50 epochs. Model performance was evaluated using accuracy, precision, recall, F1-score, ROC-AUC, and a confusion matrix.

### C. Comparative Analysis

This section relates the recommended approach to several current ones, such as Genetic Algorithm-Based Reinforcement Learning (GARL) [21], IoT–fog–cloud (IFC) [16], Intelligent Buffalo-based Secure Edge-enabled Computing (IB-SEEC) [25], Advanced Random Forest (ARF) [24] and the proposed approach measures its efficiency using performance metrics such as Response Time (ms), Throughput (Mbps), Attack detection (%), False Positive rate and Energy Consumption (%).

For the network simulation experiments, the baseline methods were implemented in NS-3.35 under the same topology, traffic model, and hardware settings summarized in Tables I and II. For the intrusion-detection experiments, all methods were trained and tested on the same CICIDS2017 partitions using identical preprocessing, feature selection, and evaluation metrics.

*1) IoT devices load (Mbps) vs. response time (ms)*: The correlation between Response Time (ms) and Total Load (Mbps) in an intelligent IoT environment can be described as:

$$RTS = \frac{TL}{AB\text{-}Uz} \tag{43}$$

where,

$RTS$ = Response Time (ms).

$TL$ = Total Load (Mbps) (data arrival rate).

$AB$ = Available Bandwidth (Mbps).

$Uz$ = Current System Utilization (Mbps).

With higher Total Load ( $TL$ ) or reduced Available Bandwidth ( $AB$ ), the Response Time $RTS$ grows with congestion and processing delays. Keeping $Uz$ low ensures quicker response times by avoiding system saturation.

TABLE III. NUMERICAL OUTCOMES OF RESPONSE TIME (MS)

| (x-axis) – Total Load of IoTdevices(Mbps) | Response Time (ms)- (y-axis) | | |
|---|---|---|---|
| | IFC | ARF | Proposed |
| 100 | 85 | 75 | 65 |
| 300 | 95 | 90 | 85 |
| 500 | 130 | 110 | 100 |
| 700 | 120 | 110 | 105 |
| 900 | 140 | 130 | 120 |

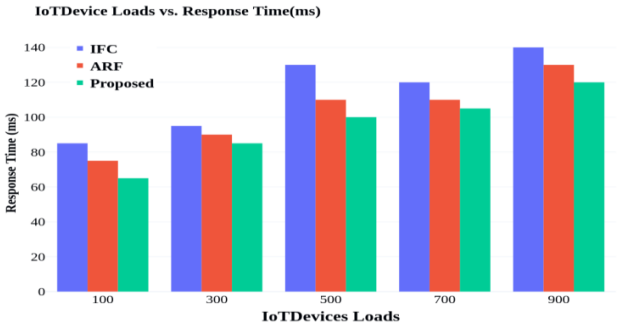


Fig. 4. IoTdevices vs. Response time (ms).

The numerical results display the response time (in ms) of three models, IFC, ARF, and the proposed, for different loads of IoT devices (in Mbps) in Fig. 4 and Table III. For a 100 Mbps load, response times are 85 ms for IFC, 75 ms for ARF, and 65 ms for the proposed method, with a 23.5% reduction from IFC. As the load is increased to 300 Mbps, the proposed method has improved efficiency at 85 ms, as opposed to 95 ms (IFC) and 90 ms (ARF). At 500 Mbps, the proposed method's response time is 100 ms, 23% quicker than IFC (130 ms). For 700 Mbps, the proposed method has 105 ms, representing a 12.5% improvement over IFC (120 ms). At the maximum load of 900 Mbps, the proposed technique reaches 120 ms with a substantial decrease in delay from IFC (140 ms) and ARF (130 ms), validating its better performance at all levels of load.

*2) Total load (Mbps) vs. Throughput (Mbps):* The correspondence between Total Load (Mbps) and Throughput (Mbps) within a smart IoT system can be expressed as:

$$Th = TL \times (1 - PL) \quad (44)$$

where,

*Th* is the Throughput (Mbps) (the actual data successfully delivered).

L is the Total Load (Mbps) (rate of incoming data).

P is the Packet Loss Ratio (ratio of lost packets).

As the Total Load *TL* rises, throughput *Th* rises in a linear manner until the system is fully utilized. After that, congestion creates packet loss *PL*, diminishing the effective throughput.

TABLE IV. NUMERICAL OUTCOMES OF THROUGHPUT (MBPS)

| (x-axis) – IoTDevices Load (Mbps) | Throughput (Mbps)- (y-axis) | | |
|---|---|---|---|
| | IB-SEEC | ARF | Proposed |
| 100 | 95 | 97 | 99 |
| 300 | 270 | 285 | 297 |
| 500 | 445 | 470 | 492 |
| 700 | 615 | 648 | 685 |
| 900 | 790 | 835 | 878 |

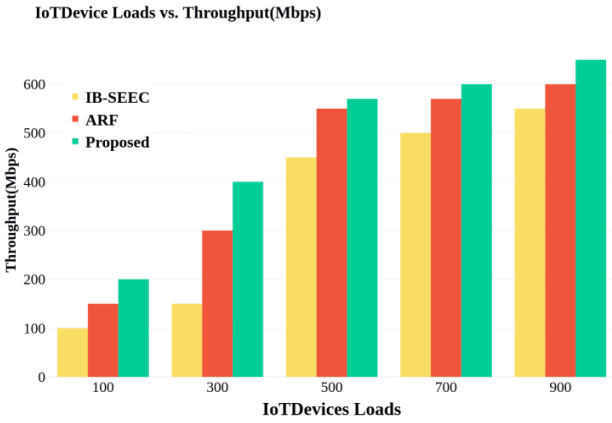


Fig. 5. IoTDevices loads vs. Throughput (Mbps).

The numerical results show the throughput (in Mbps) of three models, IB-SEEC, ARF, and the proposed method, for various IoT device loads (in Mbps) in Fig. 5 and Table IV. For a 100 Mbps load, the throughput of IB-SEEC is 95 Mbps, ARF is 97 Mbps, and the proposed method is 99 Mbps. At 300 Mbps load, the proposed method achieves 297 Mbps, compared with 270 Mbps for IB-SEEC and 285 Mbps for ARF. At 500 Mbps, the proposed method reaches 492 Mbps, outperforming IB-SEEC (445 Mbps) and ARF (470 Mbps). At 700 Mbps, the proposed method delivers 685 Mbps, compared with 615 Mbps for IB-SEEC and 648 Mbps for ARF. At the peak load of 900 Mbps, the proposed method achieves 878 Mbps, while IB-SEEC and ARF achieve 790 Mbps and 835 Mbps, respectively, confirming the superior throughput performance of the proposed method across all load levels.

*3) Number of epochs vs. Attack detection (%):* The correlation between Number of epochs and Attack detection (%) in an intelligent IoT framework based on AI is:

$$AD(\%) = AD_{max} \times (1 - e^{-\lambda NE}) \quad (45)$$

where,

*AD(%)* is Attack Detection Rate (percentage of attacked cases detected).

$AD_{max}$ is the Maximum possible Detection Accuracy by the model.

*NE* is the number of epochs (number of iterations taken during training).

$\lambda$ is Learning Rate Constant (accounts for how rapidly accuracy increases with epochs).

As the Number of epochs (*NE*) grows, the Attack Detection Rate closes in on its maximum limit $AD_{max}$, and increasing epochs produces smaller and smaller improvements.

TABLE V. NUMERICAL OUTCOMES OF ATTACK DETECTION (%)

| (x-axis) –Number of epochs | Attack detection (%)- (y-axis) | | |
|---|---|---|---|
| | GARL | ARF | Proposed |
| 10 | 15 | 25 | 35 |
| 20 | 20 | 30 | 45 |
| 30 | 40 | 55 | 65 |
| 40 | 68 | 75 | 85 |
| 50 | 75 | 85 | 95 |

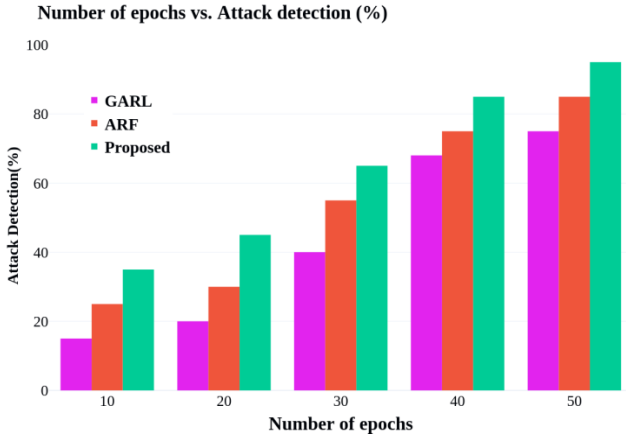


Fig. 6. Number of epochs vs. Attack detection (%).

The numerical results display the accuracy of attack detection (in %) of three models, namely, GARL, ARF, and the proposed method, on higher epochs in Fig. 6 and Table V. The proposed method performs 35% at 10 epochs, which is better than GARL (15%) and ARF (25%). As the epochs reach 20, the proposed method performs 45%, keeping a huge margin over GARL (20%) and ARF (30%). At 30 epochs, the proposed method is 25% more accurate than GARL (40%) and 10% more accurate than ARF (55%) at 65% accuracy. At 40 epochs, the proposed method still surpasses GARL (68%) and ARF (75%)

with 85% detection. At 50 epochs, the proposed method has the highest detection rate of 95%, outperforming GARL (75%) and ARF (85%), proving better accuracy and convergence speed at all epochs.

*4) Number of epochs vs. false positive rate:* The correlation between the Number of epochs and the False Positive Rate (FPR) in a smart IoT system with AI can be expressed as:

$$FPR = FPR_0 \times e^{-\varphi NE} \quad (46)$$

where,

$FPR$ is the False Positive Rate (%).

$FPR_0$ is the Initial False Positive Rate at initial training.

$NE$ is the Number of epochs (iterations during training).

$\varphi$ is a Decay Constant (specifies how rapidly the false positive rate reduces).

As the number of epochs ($NE$) gets larger, the False Positive Rate lowers exponentially because the learning is becoming better, yet the decrease occurs at a slowing rate as it converges.

TABLE VI. NUMERICAL OUTCOMES OF FPR

| (x-axis) –Number of epochs | FPR- (y-axis) | | |
|---|---|---|---|
| | GARL | ARF | Proposed |
| 10 | 0.08 | 0.07 | 0.05 |
| 20 | 0.07 | 0.06 | 0.04 |
| 30 | 0.06 | 0.05 | 0.035 |
| 40 | 0.05 | 0.04 | 0.03 |
| 50 | 0.04 | 0.03 | 0.02 |

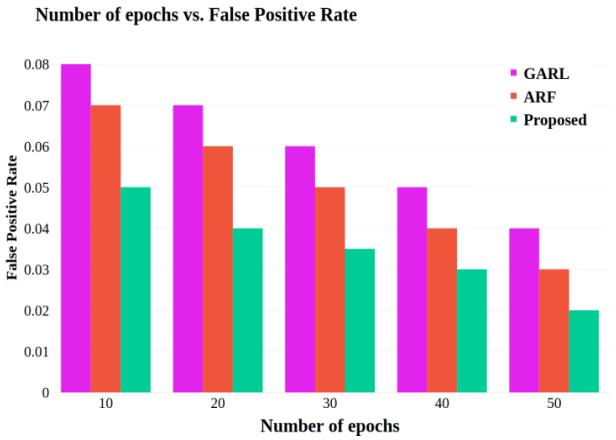


Fig. 7. Number of epochs vs. False positive rate.

The numerical results indicate the False Positive Rate (FPR) of three methods, GARL, ARF, and the proposed approach, against higher epochs in Fig. 7 and Table VI. With 10 epochs, the proposed approach has the lowest FPR of 0.05 compared to GARL (0.08) and ARF (0.07). With increased epochs to 20, the proposed approach also decreases the FPR to 0.04, whereas GARL and ARF have higher rates of 0.07 and 0.06, respectively. At 30 epochs, the proposed approach notes an FPR of 0.035, which is lower than GARL (0.06) and ARF (0.05). At 40 epochs, the proposed approach is still ahead with an FPR of 0.03, as against GARL (0.05) and ARF (0.04). For 50 epochs, the proposed model has the minimum FPR of 0.02, registering a considerable decline compared to GARL (0.04) and ARF (0.03), proving to be better in reducing false positives for all epochs.

*5) Number of IoT devices vs. Energy consumption (%):* The dependency between the Number of IoT devices and Energy Consumption (%) in a smart IoT system can be represented as:

$$EC = EC_{base} + NI \times PC_D \times AT \quad (47)$$

where,

$EC$ is the Total Energy Consumption (%).

$EC_{base}$ is the Base Energy Consumption (%) (System's idling energy consumption).

$NI$ is the Number of IoT devices.

$PC_D$ is the Power consumption per device (%).

$AT$ is the Active time (hours) of each IoT device.

TABLE VII. NUMERICAL OUTCOMES OF ENERGY CONSUMPTION (%)

| (x-axis)- Number of IoT devices | Energy Consumption (%)- (y-axis) | | |
|---|---|---|---|
| | IB-SEEC | ARF | Proposed |
| 10 | 0.9 | 0.8 | 0.5 |
| 20 | 1.0 | 0.7 | 0.6 |
| 30 | 1.3 | 0.9 | 0.8 |
| 40 | 1.4 | 1.0 | 0.9 |
| 50 | 1.5 | 1.3 | 1.0 |

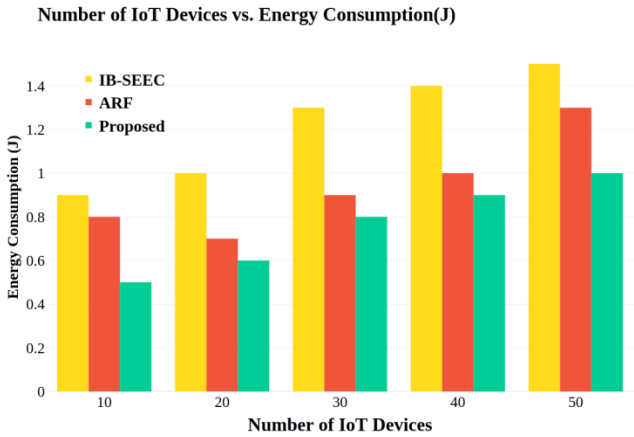


Fig. 8. Number of IoT devices vs. Energy consumption (%).

The numerical results show the energy usage (in %) of three methods, IB-SEEC, ARF, and the proposed method, over various numbers of IoT devices in Fig. 8 and Table VII. For 10 devices, the proposed method reflects the lowest energy usage at 0.5%, in contrast to IB-SEEC (0.9%) and ARF (0.8%). When the number of devices grows to 20, the proposed method reflects higher efficiency at 0.6%, as opposed to IB-SEEC and ARF with their 1.0% and 0.7%, respectively. At 30 devices, the proposed method reports 0.8%, which is lower than IB-SEEC (1.3%) and ARF (0.9%). At 40 devices, the proposed method reports 0.9%,

which is less than IB-SEEC (1.4%) and ARF (1.0%). At 50 devices, the proposed method is still the most effective at 1.0%, surpassing IB-SEEC (1.5%) and ARF (1.3%), with persistent energy savings at all levels of IoT devices.

To isolate the energy-saving contribution of Linear Network Coding (LNC), an ablation study was conducted by comparing three blockchain configurations under identical NS-3.35 settings: standard Hyperledger Fabric, Hyperledger Fabric with LNC, and the proposed integrated framework. At 50 IoT devices, energy consumption was 1.4% for standard Hyperledger Fabric, 1.1% for Hyperledger Fabric with LNC, and 1.0% for the proposed method. These results show that LNC reduces blockchain energy consumption by approximately 21.4% relative to standard Hyperledger Fabric, while the full proposed framework provides additional savings through coordinated load balancing and congestion control.

### D. Research Summary

This study suggests an intelligent IoT architecture for a green city by incorporating Artificial Intelligence (AI) and information systems for maximizing resource management, improving public services, and urban efficiency. The research tackles major challenges like inefficient feature selection and extraction, load imbalance, Quality of Service (QoS) problems, and excessive blockchain energy consumption. The suggested smart IoT network includes 50 IoT devices, 2 base stations, 1 blockchain node, and 3 servers, with data gathered using the CICIDS2017 dataset. Data pre-processing is done using the Normalization technique to normalize data. Feature optimization is done through Wrapper-based Feature Selection and Autoencoder-based Feature Extraction, enhancing data relevance and model efficiency. QoS is ensured by a mechanism of dynamic load balancing via Software Defined Networking (SDN), and stateless Q-learning is applied to avoid congestion in data transmission. To minimize energy usage in blockchain functions, Linear Network Coding (LNC) is integrated into Hyperledger Fabric for minimizing energy usage in blockchain functions. A Quasi-Recurrent Neural Network (QRNN) with optimization from the Butterfly Optimization Algorithm (BOA) is utilized to detect cyber-attacks and improve detection rates, and reduce false positives. For execution, we utilize the ns3 Python simulation tool, and we also compare the graphs that were represented in Tables IV to VII, and Fig. 4 to 8. Our techniques' performance is investigated using numerical analysis, demonstrating that it performs better across all measures.

## VI. Discussion

The NS-3.35 simulations modeled a smart-city IoT network with 50 devices, 2 base stations, 3 servers, and 1 blockchain node. Traffic was generated using a mixed UDP/TCP model with variable packet sizes and arrival rates ranging from 100 to 900 Mbps. Attack scenarios included probe, denial-of-service, and botnet traffic patterns derived from CICIDS2017 attack categories. Although the simulation scale is smaller than city-wide deployments, it provides a controlled environment for comparing load balancing, congestion control, and energy consumption trends.

Although stateless Q-learning does not maintain long-term historical state, it preserves short-term network feedback at each decision step through current utilization, queue length, and reward signals. This design reduces memory requirements while still enabling adaptive routing under changing traffic conditions.

The proposed framework indirectly supports urban sustainability by reducing response time, increasing throughput, lowering energy consumption, and improving cyber-attack detection accuracy. These improvements can support faster emergency response, more efficient resource usage, and more reliable public services in smart-city environments.

The integration of multiple intelligent modules introduces additional computational overhead compared with simpler single-module approaches. However, the proposed design mitigates this cost through feature selection, autoencoder-based dimensionality reduction, and efficient QRNN inference. The resulting performance gains in latency, throughput, security, and energy efficiency justify the added implementation complexity for city-scale IoT deployments.

Data privacy is preserved through Hyperledger Fabric authentication, encrypted transaction records, access control policies, and selective sharing of validated security events across system components. Sensitive raw traffic data remains localized within preprocessing and detection modules, while only authorized metadata and alerts are transmitted to the blockchain layer.

Failure resilience is supported through module-independent operation paths. If the blockchain layer becomes temporarily unavailable, attack detection and SDN-based traffic management continue using local logs and cached policies. If the attack detector fails, network traffic can still be routed using SDN and Q-learning rules, while blockchain logging resumes after service restoration.

Declaration of Generative AI Use: Generative AI tools were used only for grammar checking and minor language refinements during manuscript preparation. All research design, experiments, simulations, data analysis, and scientific conclusions were conducted manually by the authors.

## VII. Conclusion

The suggested smart IoT framework effectively merges AI and information systems to enhance the sustainability and efficiency of smart city operations. Through the mitigation of key challenges like poor feature selection, inefficient load balancing, and excessive energy consumption, the framework guarantees efficient resource management, improved service delivery, and decreased system congestion. The integration of Wrapper-based Feature Selection and Autoencoder enhances data processing efficiency, while the SDN-based QoS mechanism and stateless Q-learning efficiently handle real-time load distribution and congestion prevention. In addition, the implementation of Hyperledger Fabric blockchain with Linear Network Coding (LNC) minimizes energy consumption, making blockchain operations sustainable. The combination of QRNN with BOA improves attack detection accuracy and reduces false positive rates, thus enhancing the security of IoT-based smart city systems. Performance analysis shows that the proposed system offers improved response times, increased

throughput, and lower energy consumption, thus making it a suitable solution for future sustainable smart city deployments. In summary, the work offers a robust, scalable, and effective framework that can be implemented to address the rising demands and challenges in smart city settings. Future research can extend the system to multi-city networks, enhance interoperability, and integrate more advanced AI models to achieve even higher efficiency and flexibility.

## REFERENCES

[1] Auwalu, F. K., & Bello, M. (2023). Exploring the contemporary challenges of urbanization and the role of sustainable urban development: a study of Lagos City, Nigeria. *Journal of Contemporary Urban Affairs*, *7*(1), 175-188.

[2] Ali, I., & Rahman, A. (2024). Environmental Degradation: Causes, Effects and Solutions. International Journal for Multidisciplinary Research, 6(3), 1-10.

[3] Van Hoang, T. (2024). Impact of integrated artificial intelligence and internet of things technologies on smart city transformation. Journal of Technical Education Science, 19(Special Issue 01), 64-73.

[4] Stana, A., Toti, L., Kosova, R., & Prodani, F. (2019). Future of Durres: Smart City and Smart University. European Journal of Engineering and Technology Vol, 7(6).

[5] Barolli, A., Sakamoto, S., Bylykbashi, K., & Barolli, L. (2022). A hybrid intelligent simulation system for building IoT networks: Performance comparison of different router replacement methods for WMNs considering stadium distribution of IoT devices. Sensors, 22(20), 7727.

[6] Barolli, A., Sakamoto, S., Ozera, K., Barolli, L., Kulla, E., & Takizawa, M. (2018, February). Design and implementation of a hybrid intelligent system based on particle swarm optimization and distributed genetic algorithm. In International Conference on Emerging Internetworking, Data & Web Technologies (pp. 79-93). Cham: Springer International Publishing.

[7] Rani, S., Mishra, R. K., Usman, M., Kataria, A., Kumar, P., Bhambri, P., & Mishra, A. K. (2021). Amalgamation of advanced technologies for sustainable development of smart city environment: A review. *IEEE Access*, *9*, 150060-150087.

[8] Hsu, C. H., Eshwarappa, N. M., Chang, W. T., Rong, C., Zhang, W. Z., & Huang, J. (2022). Green communication approach for the smart city using renewable energy systems. *Energy Reports*, *8*, 9528-9540.

[9] Dias, T., Fonseca, T., Vitorino, J., Martins, A., Malpique, S., & Praça, I. (2023, July). From data to action: Exploring AI and IoT-driven solutions for smarter cities. In *International Symposium on Distributed Computing and Artificial Intelligence* (pp. 44-53). Cham: Springer Nature Switzerland.

[10] Stana, A., Golgota, A., & Toti, L. (2025). E-Mobility: First Step Towards the Future of a Smart City. Interdisciplinary Journal of Research and Development, 12(2 S1), 22-22.

[11] Ahmed, I., Zhang, Y., Jeon, G., Lin, W., Khosravi, M. R., & Qi, L. (2022). A blockchain-and artificial intelligence-enabled smart IoT framework for sustainable city. *International Journal of Intelligent Systems*, *37*(9), 6493-6507.

[12] Badidi, E. (2022). Edge AI and blockchain for smart sustainable cities: Promise and potential. Sustainability, 14(13), 7609.

[13] Das, D. K. (2025). Digital Technology and AI for Smart Sustainable Cities in the Global South: A Critical Review of Literature and Case Studies. *Urban Science*, *9*(3), 72.

[14] Das, D. K. (2024). Exploring the symbiotic relationship between digital transformation, infrastructure, service delivery, and governance for smart sustainable cities. Smart Cities, 7(2), 806-835.

[15] Simonofski, A., Vallé, T., Serral, E., & Wautelet, Y. (2021). Investigating context factors in citizen participation strategies: A comparative analysis of Swedish and Belgian smart cities. International Journal of Information Management, 56, 102011.

[16] Berigüete, F. E., Santos, J. S., & Rodriguez Cantalapiedra, I. (2024). Digital revolution: emerging technologies for enhancing citizen engagement in urban and environmental management. *Land*, *13*(11), 1921

[17] Ejaz, U., Ramon, W., & Olaoye, G. (2025). The Role of Big Data and AI in Smart Cities and Urban Planning.

[18] Neoaz, N. (2025). Internet of Things (IoT) and Smart Cities Examine how IoT technologies can improve urban living and infrastructure management. *Author Nahid Neoaz*.

[19] Dikshit, S., Atiq, A., Shahid, M., Dwivedi, V., & Thusu, A. (2023). The use of artificial intelligence to optimize the routing of vehicles and reduce traffic congestion in urban areas. *EAI Endorsed Transactions on Energy Web*, *10*, 1-13.

[20] Ismagilova, E., Hughes, L., Rana, N. P., & Dwivedi, Y. K. (2022). Security, privacy and risks within smart cities: Literature review and development of a smart city interaction framework. *Information Systems Frontiers*, 1-22.

[21] Mohammed, A. (2022). Cybersecurity in Smart Cities: Securing IoT and Smart Infrastructure. Journal of Innovative Technologies, 5(1).

[22] Alzyoud, F., Al-Falah, R., Tarawneh, M., & Tarawneh, O. (2024, March). Security challenges and solutions in smart cities. In International Conference on Advances in Computing Research (pp. 256-267). Cham: Springer Nature Switzerland.

[23] Vijarania, M., Gupta, S., Agrawal, A., Adigun, M. O., Ajagbe, S. A., & Awotunde, J. B. (2023). Energy efficient load-balancing mechanism in integrated IoT–fog–cloud environment. Electronics, 12(11), 2543.

[24] Yuvaraj, N., Praghash, K., Logeshwaran, J., Peter, G., & Stonier, A. A. (2023). An artificial intelligence based sustainable approaches—IoT systems for smart cities. In AI Models for Blockchain-Based Intelligent Networks in IoT Systems: Concepts, Methodologies, Tools, and Applications (pp. 105-120). Cham: Springer International Publishing.

[25] Nagarajan, S. M., Deverajan, G. G., Chatterjee, P., Alnumay, W., & Muthukumaran, V. (2022). Integration of IoT based routing process for food supply chain management in sustainable smart cities. Sustainable Cities and Society, 76, 103448.

[26] Jannu, S., Dara, S., Thuppari, C., Vidyarthi, A., & Gupta, D. (2023). An advanced energy management and harvesting system for network lifetime for industrial IoT in smart cities. IEEE Internet of Things Journal.

[27] Wen, W., Demirbaga, U., Singh, A., Jindal, A., Batth, R. S., Zhang, P., & Aujla, G. S. (2023). Health monitoring and diagnosis for geo-distributed edge ecosystem in smart city. IEEE Internet of Things Journal, 10(21), 18571-18578.

[28] Ajao, L. A., & Apeh, S. T. (2023). Secure edge computing vulnerabilities in smart cities sustainability using petri net and genetic algorithm-based reinforcement learning. Intelligent Systems with Applications, 18, 200216.

[29] Khadidos, A. O., Shitharth, S., Manoharan, H., Yafoz, A., Khadidos, A. O., & Alyoubi, K. H. (2022). An intelligent security framework based on collaborative mutual authentication model for smart city networks. IEEE Access, 10, 85289-85304.

[30] Luo, J., Zhao, C., Chen, Q., & Li, G. (2022). Using deep belief network to construct the agricultural information system based on Internet of Things. The Journal of Supercomputing, 78(1), 379-405.

[31] Kolhe, R. V., William, P., Yawalkar, P. M., Paithankar, D. N., & Pabale, A. R. (2023). Smart city implementation based on Internet of Things integrated with optimization technology. Measurement: Sensors, 27, 100789.

[32] Faisal, M., Sahabuddin, Hasiri, E. M., Darniati, Abd Rahman, T. K., Mulyadi, I., & Widia, I. D. M. (2025). Assessing AI-Driven Personalization in Smart Cities Using Hybrid Machine Learning and MCDM Approach. HighTech and Innovation Journal, 6(3), 770–792.

[33] Huy, P. Q., & Phuc, V. K. (2025). Impact of Technologically Vigilant Leadership on Smart Sustainable Circular Supply Chain Management. Emerging Science Journal, 9(4), 1868–1885.

[34] Xiao, Y., & Dong, Y. (2025). Optimizing AIGC Technology for IoT Devices with Deep Learning. HighTech and Innovation Journal, 6(3), 976–990.