International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

A Fog-Enabled OMNeT++ Framework for Network Anomaly Detection in Smart City Environments

Akash Mehta[1], Dr. Minal Patel[2]

[1]Research Scholar, Department of Computer/IT Engineering, Gujarat Technological University, Ahmedabad, Gujarat, India, akash.mehta.it@gmail.com

[2]Associate Professor, Computer Engineering Department, Sardar Vallabhbhai Patel Institute of Technology - Vasad, Gujarat Technological University

Abstract

In the age of smart cities, maintaining safe and reliable data transmission over large-scale sensor networks is becoming more important. These networks, which are often made up of smart cameras and IoT devices, are susceptible to abnormalities like rogue nodes, sensor failures, and traffic spikes, which may jeopardize system integrity and performance. Traditional centralized anomaly detection methods suffer from latency and scalability difficulties, particularly in high-density settings. To solve this issue, this paper proposes a new anomaly detection system that combines Ant Colony Optimization (ACO) and clustering approaches in an OMNeT++ simulation environment. The system uses a fog-cloud architecture, with fog nodes doing localized processing and clustering to decrease latency and data overhead, and ACO for efficient data routing and anomaly detection. Simulations were run with three distinct sensitivity settings: baseline, high, and low, to assess detection accuracy, precision, recall, and F1-score. The suggested technique showed considerable increases in all measures, with baseline scenario accuracy improving from 86.5% (without detection) to 92.4% (with detection), and F1-score from 0.64 to 0.837. Furthermore, the system demonstrated improved processing efficiency and decreased network use at both the fog and cloud levels, indicating its applicability for real-time anomaly identification in dynamic and scalable smart city networks.

Keywords: OMNeT++, ACO, Clustering, Smart City Networks, Fog Computing.

1 Introduction

The emergence of the Internet of Things (IoT) is seen by many experts as a major breakthrough in information and communication technology (ICT) after the creation of computers and the Internet. The IoTs may connect a variety of smart devices and sensors to the network, allowing for the collection of data from the actual world. This feature makes it possible to store and handle the collected data in an automated and dynamic manner [1], [2]. The rise of IoT-powered smart cities has fundamentally altered the way how the cities function and expand.

To increase operational effectiveness, sustainability, and the general quality of life for their residents, these cities make use of digital connections and a vast sensor network [3]. However, an unprecedented amount of data has entered urban surroundings as a result of the growth of IoT devices. Although this data is

essential to the real-time operations and decision-making processes of a smart city, it also poses serious difficulties, especially in the areas of administration and security [4]. Finding abnormalities in big urban data is one of the most important difficulties.

A security breach in a smart city may have a significant effect on economic stability, governmental services, and community confidence. Data leaks, denial of service (DoS) attacks, ransomware, phishing, and IoT device manipulation are typical cyberthreats in smart cities [5]. Research and development efforts have been focused on strengthening smart cities' cybersecurity posture because of the severity of these threats. Firewalls, encryption mechanisms, and intrusion detection systems are examples of current solutions. But these conventional methods often fall short in tackling the particular problems faced by smart cities [6].

The goal of anomaly detection is to find trends or occurrences that substantially depart from the typical or anticipated behavior of a data collection. In the case of smart cities, this job is crucial since it encompasses a wide range of applications, from seeing electrical grid manipulation to spotting cyberthreats and predicting malfunctions in vital urban systems [7] , [8]. In addition to being beneficial, early anomaly detection is essential for averting unforeseen disruptions, protecting public safety, and preserving the integrity of urban systems. IoT devices act as the nerve ends of a smart city's complex fabric, continually monitoring and gathering data from a variety of urban living factors, such as energy use, intelligent transportation systems, environmental conditions, and public security. As a result, the growth of IoT in smart cities has created massive amounts of data, which have put a strain on the network and processing power. These massive amounts of data require a lot of computational power, communication bandwidth, storage space and real time threat detection. Handling the massive amounts of data generated by IoT applications is proving to be a significant problem for the academic and scientific community [9], [10]. This challenge highlights the need for innovative solutions such as data-proxy anomaly detection in smart cities.

In response to this problem, the Cloud of Things (CoT) emerged as a result of the convergence of cloud computing with IoT [11]. Furthermore, cloud computing offers a centralized computing approach with a large amount of storage and processing power. This connection makes it possible to collect data from IoT devices in a seamless manner and streamlines the processing of the collected data [12]. Therefore, the CoT model, in which devices send data straight to the cloud, is shown in Figure 1. Following that, a suitable choice is made based on the findings of the analysis and calculation, which are both done in the cloud.

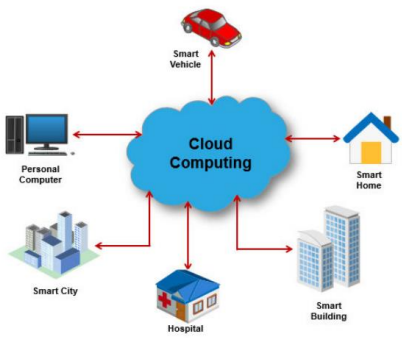


Figure 1 The communication model of the Cloud of Things

Among the many advantages of this strategy is the low maintenance and monitoring requirements. It has thus created a multibillion-dollar business. However, centralized cloud-based models for anomaly

detection present notable limitations such as increased network jitter, higher bandwidth consumption, scalability issues and latency concerns related to data privacy [13], [14]. These constraints limit the efficacy of timely threat detection and rapid incident response in smart city environments, where real-time decision-making is crucial.

To address these challenges, Cisco has created the fog computing wherein computational resources are transferred to the network edge. Fog computing minimizes resource contention at the edge and maximizes resource usage by coordinating the use of geographically dispersed network edge devices and using cloud resources. Because of this, fog computing is able to balance resource use and enhance overall resource efficiency [15].

By bringing computing resources closer to the network's edge, fog computing allows real-time threat response and detection with decreased latency and network load. Fog computing decouples the decentralized method to offer faster processing of edge device-created data, making possible real-time anomaly detection as well as local mitigation measures in smart city environments. Figure 2 demonstrates the basic architecture of fog computing, consisting of the three-layers.

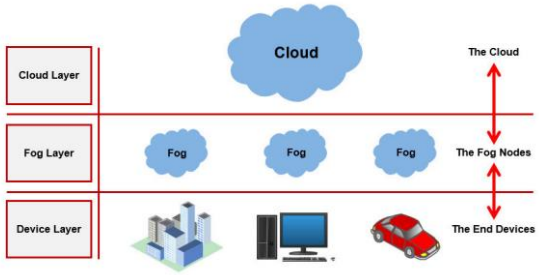


Figure 2 Fog Computing Architecture

Despite its advantages, anomaly detection system development and testing in fog scenarios is still a challenging task given the complexity, heterogeneity, and size of smart city networks. Real-world testing and deployment are costly and mostly impractical. Hence, simulation-based evaluation frameworks that can support controlled, repeatable, and scalable detection algorithm testing in realistic conditions are increasingly in demand.

An extendable modeling library for creating network simulations is called OMNeT++. It is free for non-commercial and academic usage and is an open source product. Model component architecture is provided by OMNeT++. C++ is used to program the components (modules), which are then put together using the NED language to create bigger components and models [16]. The integrated development environment in OMNeT++ offers a wealth of features for examining models. INET is the most widely used modeling library that extends OMNeT++. It offers agents, protocols, and more models for data network operations [17].

OMNeT++ provides a modular, flexible simulation framework well suited to model fog-based networks. By combining with optimization algorithms such as Ant Colony Optimization (ACO) for effective resource allocation and clustering algorithms for pattern analysis of behavioral patterns, OMNeT++ is an effective platform for developing adaptive, scalable, and high-accuracy anomaly detection systems. This

research is thus motivated by the need to create a fog-enabled OMNeT++ framework that supports real-time, intelligent threat detection tailored to the dynamic and distributed nature of smart city ecosystems.

The key contributions of this research include:

- To develop a fog-enabled anomaly detection framework using OMNeT++ that supports real-time, distributed threat identification in smart city IoT environments.
- To enhance the efficiency and accuracy of anomaly detection by integrating intelligent algorithms such as Ant Colony Optimization (ACO) and clustering techniques within the simulation.
- To demonstrate the advantages of fog computing over traditional cloud-based approaches, including reduced latency, lower bandwidth consumption, and improved scalability for large-scale urban networks.
- To provide a privacy-aware, simulation-based testing environment that overcomes the cost and complexity of real-world deployments, enabling reproducible and realistic evaluation of smart city cybersecurity solutions.

The remaining sections will be structured as follows: Section 2 delves into the associated work, which involves studying the literature for our suggested approach and identifying any gaps in the research. Section 3 provides background information on the suggested models. Section 4 covers the suggested methodology, which includes a data collection description of the dataset, preprocessing, segmentation, and model. Section 5 explains the experiment, analyzes the results, and compares them. Section 6 presents the study's conclusions and future directions.

2 Literature Review

This section reviews the literature with a focus on fog computing simulation models, optimization techniques, and IoT-based systems. Current methods for anomaly detection in smart city environments are explored, and significant research is highlighted.

Anomaly detection in IoT and smart city networks

Urban security benefits greatly from big data, but integrating these data systems is a challenging task. The complex data management procedure for big data integration that occurs when smart cities combine datasets from cloud platforms, edge computing systems, and Internet of Things devices is described by [18]. For security professionals, integrating many data sources seamlessly and accurately maintaining data dependability is a major difficulty. According to [19], big data helps researchers create comprehensive risk assessment models that look at environmental factors, system vulnerability traits, and past security occurrences. [20] explains how big data allows data-driven judgments by merging data from cybersecurity monitoring devices, emergency units, social media feeds, and surveillance systems. Administrators can identify security threats, initiate emergency responses, and improve police protocols by analyzing this data.

A smart city is susceptible to a number of sophisticated cyberattacks, however. Deep autoencoder techniques are proposed by [21] to identify cyberattacks in self-driving cars. In order to facilitate data interchange and optimize vehicle operation, the research examines how electronic control units (ECUs) behave in connected and autonomous cars (CAVs), which are linked via in-vehicle networks (IVNs). Machine learning and deep learning techniques are used in the analysis to find cyberattacks that find

inaccurate data on car data buses. Gradient boosting, k-nearest neighbor (KNN), decision trees, and long short-term memory (LSTM) are a few of the techniques used in the research. A split-training technique was used by [22] to identify anomalies in gas turbine engines. To create a simulation model and extract input data, a clustering approach was used. By streamlining the whole process, a regression model improves model performance while classifying the data points.

For anomaly identification, [23] suggest a split active learning approach in conjunction with unsupervised techniques. To lower labeling expenses, the research used autoencoders with active learning. The suggested approach outperforms conventional learning techniques by cutting training time and increasing performance by 20%. A strategy for data splitting in independently and identically distributed (IID) testing splits is put out by [24]. A variety of techniques were used to examine the performance, and the receiver operating characteristic (ROC) curve was used for validation. For network anomaly detection, [25] use adaptive multiclass balancing in conjunction with semi-supervised learning. To address data imbalance, the research used an adaptive confidence threshold function and a multiclass split balancing approach. The suggested method outperforms alternative baseline models and improves anomaly detection performance.

Fog-Based Anomaly Detection in IoT

A fog computing-based IDS architecture for IoT networks was proposed by [26]. Their suggested method makes use of fog computing to detect and stop intrusions, hence enhancing network security. One of the results is an improvement in network resilience to cyberthreats. A thorough literature evaluation of IDS and prevention in fog-based IoT settings was conducted by [27]. By analyzing several intrusion detection methods, this study improves the suggested design and adds to the corpus of work by literature. They provide a more thorough perspective by looking at several approaches, as opposed to literature which focus on putting up a specific architectural concept. This study lays the groundwork for future advancements in the subject and makes it easier to identify research needs.

In order to identify attacks in IoT-fog situations, [28] suggested SIMAD, a secure and intelligent technique. They provide a logical plan to strengthen network security, building on previous studies complaint about the need for more sophisticated IDS. By improving network security and threat detection capabilities, their suggested method aims to address some of the shortcomings noted in the literature study.

A fog computing-based DDoS attack mitigation strategy for Internet of Things networks was presented by [29]. This approach expands research in IoT-fog settings by tackling specific threats like DDoS assaults. By allocating detection and mitigation tasks across fog nodes, their approach aims to increase network resilience against DDoS attacks while simultaneously enhancing network security. Moreover, [30] suggested a safe architecture for the Internet of Things based on fuzzy logic and fog. By adding fuzzy logic-based decision-making processes to this architecture, fog computing environments become more secure . They provide a security architecture that addresses a broader variety of issues in IoT systems, such as intrusion detection, in contrast to Lawal et al.'s focused attention on specific threats like DDoS assaults.

For fog-assisted IoT systems, [31] presented a secure integrated framework. Their architecture successfully addresses the security issues in IoT devices by combining fog computing capabilities with traditional security measures. This paradigm, which builds on the work of Zahra and Chishti, provides a thorough understanding of IoT security by combining traditional and fog-based security measures.

Furthermore, [32] suggested a general and lightweight security solution for identifying malicious activity in uncertain IoT using fuzzy logic and a fog-based approach. They continued to research the security implications of fuzzy logic in IoT systems despite constraints.

Simulation-Based Evaluation of Fog Computing Network Performance

The following is a discussion and review of the most relevant papers about simulation fog computing network performance: In order to produce unique patterns that aid in data movement and gauge the degree of infrastructure development, [17] relied on the development of simulation models for the fog computing infrastructure based on OMNeT++ in addition to tools for evaluating the network's condition and the dynamics of its PlayStations on a set of statistical models.

A study and architectural idea for combining IoT application development and simulation in edge and fog computing network settings was proposed by [33]. The FogNetSim++ extension and the node-RED application middleware serve as the foundation for the suggested design. The manager component of this integration is in charge of facilitating data exchange between the simulator and the middleware. The outcomes of the tests demonstrate how the integration works and how IoT applications may be validated using them.

In order to overcome some of the disadvantages of conventional cloud computing, namely its high energy consumption, [34] used network modeling toolkits. The finest network simulation program for evaluating fog computing and conventional cloud computing critically in terms of planning, data management, and energy efficiency was found via an analysis. They demonstrate that implementing fog computing layer technology lowers energy consumption in comparison to standard cloud computing architecture using the iFogSim network simulation toolbox. Additionally, the differences between fog and cloud computing data centers are examined with regard to data management and energy consumption.

According to research by [35], a technique was used to help improve the unique simulator MobFogSim, which incorporates dynamic network slicing in service management for mobile fog networks. It also helped add an empirical assessment of why network slicing in the fog environment benefits mobile device reception and may result in higher resource consumption, as well as a posting tool for requests generated from the nodes.

Optimization and clustering techniques in network security

Researchers have created a wide range of approaches, including hybrid models, clustering and optimization algorithms, to tackle the many difficulties in intrusion detection for IoT networks. Finding trends and outliers is aided by clustering, which combines related data points according to similarity or distance measurements [36]. Clustering is an unsupervised learning approach used in IoT network attack detection to group similar traffic patterns or behaviors in order to find anomalies that could point to possible assaults. Without labeled data, clustering may identify anomalous behavior and reveal hidden patterns by evaluating data according to distance or similarity metrics. By differentiating between malicious and legitimate communication, it improves intrusion detection, particularly in dynamic IoT contexts. However, the difficulty of choosing the best method, high processing costs, and sensitivity to parameter adjustment may restrict its efficacy. In large-scale IoT networks, common methods like DBSCAN [37], Clique [38] and K-means [38] are often used to find clusters of anomalous behavior. These developments demonstrate how intrusion detection techniques are constantly changing due to the

unique needs and features of Internet of Things networks as well as the ever-changing nature of network threats, emphasizing the need for flexible and creative solutions.

Regarding network security optimization techniques, [39] discovered that the ACO algorithm demonstrated clear benefits in terms of random number generation, encryption strength, encryption and decryption times, and key distribution times. This suggests that the ACO algorithm has significant potential for use in encryption systems for computer network communication security. Furthermore, for the purpose of identifying cyberattacks in smart cities, [35] proposed a unique white shark equilibrium optimizer and hybrid deep learning techniques. The goal of the research is to maximize power management in order to enhance resource efficiency and quality of life. The White Shark Equilibrium Optimizer with a Hybrid Deep-Learning-based Cybersecurity Solution (WSEO-HDLCS) is used in the research to solve the problem of DDoS assaults interfering with critical services. Although the research addresses practical implementations and scalability difficulties, and the findings seem good, the study's major emphasis is restricted to DDoS assaults. Additionally, [40] examined several performance-based AI algorithms to accurately forecast IoT device issues and assaults. Ant colony optimization, genetic algorithms, and particle swarm optimization (PSO) were used to illustrate the efficacy of the proposed method with respect to four distinct parameters. The suggested approach using PSO produced results that were around 73% better than those of the current systems.

2.1 Research gap

Despite extensive efforts in designing anomaly detection systems for IoT-based smart city networks, existing solutions tend to be non-real-time responsive, scalable, and adaptable to dynamic distributed environments. Most existing solutions heavily depend on centralized cloud infrastructures, which bring latency, bandwidth limitations, and possible privacy issues. While fog computing and intelligent algorithms such as ACO and clustering have proven potential separately, there exists a significant lack of research integrating these methods in a simulation-based environment such as OMNeT++ to model, analyze, and test adaptive, privacy-preserving, and resource-conscious anomaly detection systems specifically designed for smart cities. This research intends to bridge this gap by designing a fog-based OMNeT++ platform integrating optimization and clustering methods for real-time threat detection and analysis.

3 Background

3.1 OMNeT++

OMNeT++ is a discrete event simulator built in C++ that may be used to represent multiprocessors, communication networks, and other distributed or parallel systems. Because OMNeT++ is open-source, it is allowed to be used for non-profit purposes under the terms of the Academic Public License. A robust open-source discrete event simulation tool that academic, educational, and research-focused commercial organizations may use to simulate computer networks and distributed or parallel systems was the driving force behind the development of OMNeT++. OMNeT++ is an example of a framework method. It offers the fundamental equipment and resources needed to create simulations for computer networks, queuing networks, and other areas rather than direct simulation components. Leveraging this modular and extensible architecture, researchers and developers have been able to build a wide range of simulation models for various domains, including wireless and ad-hoc networks, sensor networks, IP and IPv6 networks, MPLS, wireless channels, peer-to-peer networks, storage area networks (SANs), optical

networks, queuing networks, file systems, high-speed interconnections (InfiniBand), and others, simulation models have been created by numerous individuals and research groups since their initial release.

Model Structure

Modules that communicate via message passing make up an OMNeT++ paradigm. Using the simulation class library, the active modules—also known as simple modules—are developed in C++. The number of hierarchical levels is unlimited; simple modules may be combined into compound modules, and so on. It is possible to send messages directly to their target modules or via connections that span across modules. DEVS [41] atomic and linked models have similarities with the idea of simple and complex modules.

Module types include both basic and complicated modules. Module types are defined by the user when defining the model; instances of these module types function as building blocks for more intricate module types. In the end, the user constructs the system module as a network module, a unique kind of compound module devoid of gateways to the outside world. There is no difference between a simple and complex module when it is used as a building block. This enables the user to re-implement the functionality of a compound module in a single simple module, or to transparently divide a module into several simple modules inside a compound module, without impacting current users of the module type. Model frameworks such as the INET Framework [42]and the Mobility Framework [43], together with its expansions, demonstrate the viability of model reuse.

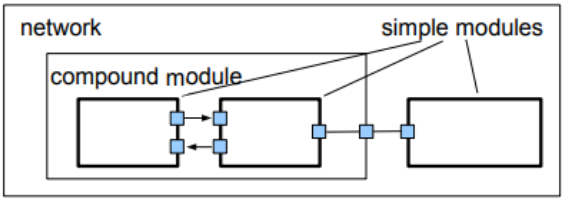


Figure 3 *Model Structure of OMNeT++*

Module messages may include arbitrary data in addition to standard features like timestamps. While simple modules normally transmit messages via gates, they may also send them directly to their target modules. Modules have input and output interfaces, with messages transiting via output and input gates. An input and output gate may be connected. Connections may be made inside a single module hierarchy level, such as connecting gates of two submodules or one submodule and a compound module gate. To ensure model reuse, connections across hierarchy levels are not allowed. Messages generally pass along a chain of links in the hierarchical architecture, starting and arriving in basic modules. Compound modules behave as 'cardboard boxes' in the model, transparently transmitting messages between internal and external components.

Properties like propagation delay, data throughput, and bit error rate may be applied to connections. Connection types with certain features (called channels) may be defined and reused in several locations. A module may contain parameters. Parameters are mostly used to send configuration data to basic modules and establish model topology. Parameters may be textual, numeric, or Boolean. As program objects, parameters can represent constants, generate random numbers, prompt the user for values, and store expressions referencing other parameters. Compound modules may transmit parameters or expressions to submodules.

3.2 Ant Colony Optimization Algorithm

Ant colony optimization (ACO) is a metaheuristic optimization approach that relies on the behaviour of ants. The development of this technology was place in the early 1990s by Dorigo [44]. The first concept stems from the study of ants' use of food resources. While ants may have low individual cognitive ability, they possess the collective intelligence to efficiently locate the quickest route between a food source and their colony.

When searching for food, ants first explore the area around their nest in a random manner. If an ant discovers a food supply, it assesses it and returns some food to the nest. During its return voyage, the ant leaves a pheromone trail on the route taken. The deposited pheromone is proportional to the amount and quality of the food, and it directs other ants to this source. Pheromone trails constitute indirect communication (also known as stigmergy) among ants, helping them to locate the shortest distance between their colony and food sources. This method is shown in Figure 4.

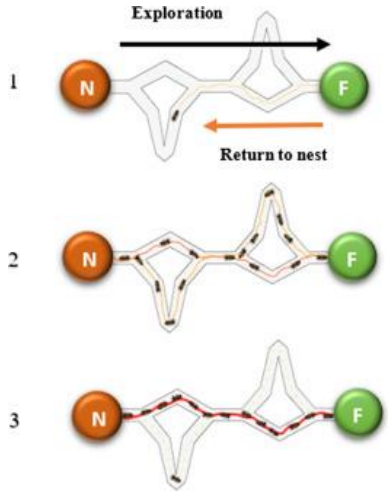


Figure 4 Ant colony choose the quickest way. An ant discovers a food source (F) and returns to its nest (N) by dispersing a pheromone along its route [45]**.**

The artificial ACO uses genuine ant colony capabilities to approximate optimization solutions. Although the first version of ACO only applies to discrete domains, the pheromone method has now been expanded to include continuous domains [46]. This is accomplished by creating a so-called solution archive, which contains a list of potential solutions that lead to a Gaussian mixed probabilistic model. Because of the evolutionary interaction between the probabilistic model and solution archive, the ACO in continuous domains is also known as the Estimation of Distribution algorithm [47]. The key phases of this method are outlined as follows:

- ***Initialization:*** An initial population of n ants is formed randomly in a search area, and their fitness functions are assessed.
- ***Generation of the solution archive:*** Initial solutions are ranked based on their fitness. The best and worst solutions are indicated by x_1 and x_n, respectively.
- ***Weight Attribution:*** The solutions regrouped in the archive are assigned a weight by using the following equation:

$$w_i \infty \frac{1}{\sqrt{2\pi\alpha n}} \exp\left[-\frac{1}{2}\left(\frac{i-1}{\alpha n}\right)^2\right] \quad (1)$$

$$\sum_{i=1}^{n} w_i = 1 \quad (2)$$

- ***Generating the probabilistic model:*** The probabilistic model based on the Gaussian mixture is given by the following equation:

$$G^k(x[k]) = \sum_{i=1}^{n} w_i \, N(x[k]; \mu_i[k]; \sigma_i[k]) \quad (3)$$

$$N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (4)$$

where $x[k]$ denotes the kth element in x and k is the decision variable.

The probabilistic paradigm is implemented by computing the mean and standard deviation of the Gaussian mixture, as demonstrated in the following expressions:

$$\mu_i[k] = x_i[k] \quad (5)$$

$$\sigma_i[k] = \frac{\eta}{n-1} \sum_{l=1}^{J} \left[x_{\substack{J \\ l}}[k] \right] \quad (6)$$

Sampling: m new samples are created as offsprings of the preceding archive by utilizing g=$(G^1, G^2, \ldots, G^{n_x})$. Offspring are evaluated based on the fitness function.

Selection: In this stage, the offspring creates a new solution archive and finds the best solutions. The fittest solution in the archive is therefore the optimal solution of the optimization process. This operation is continued until a stop condition is fulfilled.

3.3 Clustering

Clustering is an unsupervised learning technique, in contrast to classification. Clustering methods have been used throughout the years in a variety of fields, including intrusion detection [48], networking [49], data mining, document analysis [50], image processing [51], and more.

K Means Clustering:

Large data sets are often clustered using the well-liked data mining clustering method K-means [52]. It was among the simplest unsupervised learning methods used to the problem with the famous cluster. The K-means Clustering is a situation in which the centroid of the cluster serves as its representation. The chosen centroids are not required to belong to the cluster. Figure 5 depicts the basic steps.

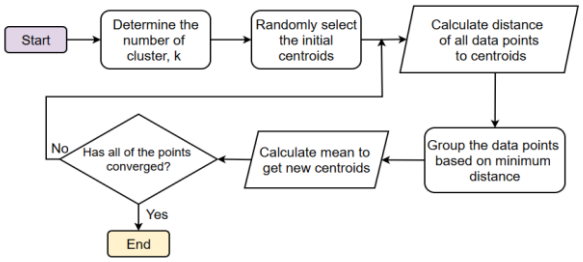


Figure 5 Basic steps of K-means Clustering

The algorithm of the K-Means works as per below steps:

- **Step 1 :** Choose the centroids for each cluster at random.
- **Step 2 :** Determine each data point's distance from the centroids, then group them into the nearest cluster.
- **Step 3 :** Calculate each cluster's new centroids by averaging all of its data points.
- **Step 4 :** Continue steps 2 and 3 until the centroids stop moving and all the points have converged.

There are two separate phases available in the algorithm. In the first stage, k centres are chosen at random; k is a predetermined number. After that, it is required to deliver every data items to the closest centre. The distance between the cluster centres and every data point is often calculated using the Euclidean distance. Early grouping and the first stage are finished when each data item belongs to a few clusters. Recalculating the early-formed clusters' average. The criteria function is iterated through until the lowest value is obtained.

The criteria function is defined as follows, considering that x is the target item and that xi is the cluster Ci average:

$$E = \sum_{i=1}^{K} \sum_{X \in C_i} \left| x - x_i \right|^2 \qquad (7)$$

For each database entry, E is the squared error total. The Euclidean distance is the length of the criteria function, which finds the shortest path between the cluster centre and each data point.

The Euclidean separation of a single vector X and another vector Y.

Where , $X = (X_1, X_2, \ldots \ldots X_n)$

$Y = (Y_1, Y_2, \ldots \ldots Y_n)$

The Euclidean gap (distance) is calculated through the following equation.

$$d(X_i, Y_i) = \left[\sum_{i=1}^{n} (x_i - y_i)^2 \right]^{1/2} \qquad (8)$$

Algorithm 1. *K*- Means Standard Algorithm

 # Initialisation
Value of $N := \{x1, \ldots, xn\}$;
Value of $M := \{\mu 1, \ldots, \mu k\}$;
 # Categorization
For $x_i \in N$ and $\mu k \in M$
Euclidean gap from every x_i to k Clustering Calculation
Designate object x_i to nearest Centroid μk
Calculation for Centroid:
Calculate Centroid μk
 # Convergence:
If $M := \{\mu 1, \ldots, \mu k\}$ remains unaffected in Two successive iterations **then:**
End the Procedure;
else

14: Go to **Categorization**
15: **End**

Density-Based Clustering

Density-based spatial clustering of applications with noise (DBSCAN), which was first presented by Rehman et al., defines clusters as high-density regions that divide from low-density regions. It can identify any kind of cluster and is outlier-tolerant and noisy [53]. It has emerged as one of the most popular and widely used clustering techniques due to its ability to recognise clusters of various shapes and handle datasets with varying densities. Depending on the dataset density in the features region, DBSCAN identifies core, border, and noise points [54]. In order for an area to be considered dense, it needs two parameters: (1) minPts, which is the smallest number of points that must be clustered together, and (2) eps, which is the distance measure that will identify the points in the vicinity of any point. Figure 6 provides an illustration of the fundamental stages.

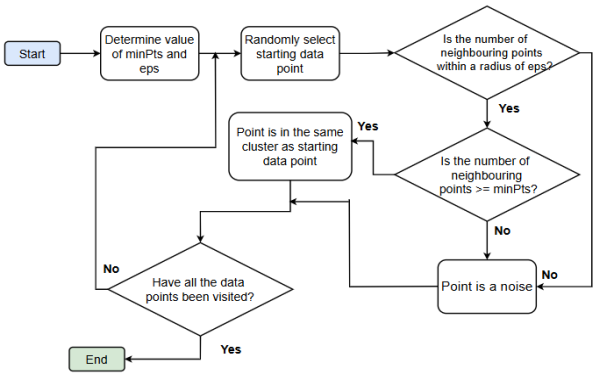


Figure 6 Basic steps of DBSCAN.

- **Step 1:** Find the minPts and eps values.
- **Step 2:** Choose a beginning data point at random. The points belong to the same cluster if there are at least minPts within an eps radius of the initial data point. If not, the point is regarded as a noise.
- **Step 3:** Continue from step 2 until every point has been reached.

Certain problems and situations are addressed by other DBSCAN versions and improvements. For instance, k-DBSCAN improves on DBSCAN by addressing the problem of dimensionality and providing guidance on feature selection and handling huge data. Using local density, the local outlier factor (LOF) [55] extension finds outliers in data. These improvements and adjustments demonstrate DBSCAN's adaptability to different clustering conditions. By integrating new techniques and modifications, these enhancements increase the algorithm's efficacy and applicability across a range of fields, including geographical analysis of data, anomaly detection, and pattern identification.

4 METHODOLOGY

This research employs a simulation-based approach to develop and test anomaly detection in a smart city network, using the OMNeT++ framework, Ant Colony Optimization (ACO), and clustering algorithms.

The process includes the following important steps:

Network Simulation (OMNeT++ + INET Framework)

The smart city network simulation is built and run on the OMNeT++ simulation environment, which is integrated with the INET framework. The network architecture consists of many components, including WirelessHosts that represent smart devices, Routers that manage data transfer, and StandardHosts that simulate static network elements such as servers and gateways. These components interact to simulate a real-world smart city infrastructure.

Simulation settings are carefully set to specify communication behaviors, IP addresses, data transmission intervals, and sensor node functioning via wireless and Ethernet connections. At the application layer, sensor-based applications such as traffic sensors, smart lighting systems, and environmental monitors are installed. These programs produce and send realistic traffic data to a centralized gateway node. The gateway takes data and sends it to a backend server, where it is recorded and processed for future analysis. This complete configuration allows for the analysis of communication patterns and the identification of abnormalities in a simulated smart city setting.

Step 1 Network Simulation Setup

1: Initialize OMNeT++ simulation environment

2: Define network topology using .ned file

3: Configure simulation parameters in omnetpp.ini

4: Deploy sensor nodes with applications:

trafficSensorApp, smartLightApp, envSensorApp

5: Define data flow to gatewayApp and serverApp

6: Start simulation and monitor network metrics

Data collection and logging

In the simulation, the dataset is created synthetically using two essential methods inside the OMNeT++ code, allowing for both regular network data production and the insertion of abnormalities. The dataset is generated in real time throughout the simulation run, guaranteeing that it accurately depicts dynamic network circumstances and behaviors.

Normal Network Data Generation

The collectNetworkData() function contains the first mechanism, which creates data points based on realistic parameters with normal distributions. Among these criteria are:

Traffic Rate: Produced using the standard deviation (normalTrafficStdDev) and mean (normalTrafficMean) of average network traffic rates.

Latency**:** Similarly, latency has a normal distribution, with mean (normalLatencyMean) and standard deviation normalLatencyStdDev) values.

Packet Loss**:** This is constructed using a uniform distribution, with values ranging from 0 to 5%.

Energy Consumption**:** Energy consumption similarly follows a uniform distribution, with values ranging from 0.5 to 1.5.

Step 2 Data Collection and Logging

1: Collect the following parameters for each time step t:

2: - Traffic Rate: TR(t)

3: - Latency: L(t)

4: - Packet Loss: PL(t)

5: - Energy Consumption: E(t)

6: Store: X(t) = [TR(t), L(t), PL(t), E(t)]

Anomaly Injection

The second approach, implemented in the inject Anomaly() function, alters the usual parameters to generate anomalous data points. The anomaly Probability option controls the likelihood of an anomaly being injected; by default, it is set to 0.08 (or 8%). Three different sorts of abnormalities are injected:

- Traffic Spike: The traffic rate increases by 2-5 times the typical rate, resulting in greater packet loss rates.
- Sensor Failure: Latency is raised by 3-8 times the typical amount, accompanied by a significant packet loss.
- Malicious Node: Traffic increases somewhat, while energy usage rises significantly.

Dataset characteristics

For a 100-second simulation, the resulting dataset comprises the following features:

- There are around 1,000 data points, and fresh data is created every 0.1 seconds.
- Based on an anomaly probability of 8%, there were around 80-90 injected abnormalities.
- Each data point includes the following attributes: timestamp, traffic rate, latency, packet loss, and energy usage.
- Ground truth labels are added to indicate whether the data point is normal or represents a certain sort of abnormality (for example, a traffic surge, sensor failure, or malicious node).

Step 3 Anomaly Injection

1: Define normal data vector: X = [TR, L, PL, E]

2: For anomaly probability p, inject anomaly if $r < p$, $r \sim U(0, 1)$

3: Modify features as:

4: Traffic Spike: $TR' = \alpha TR, \alpha \in [2, 5]$

5: Sensor Failure: $L' = \beta L, \beta \in [3, 8]$, $PL' = 100\%$

6: Malicious Node: $E' = \gamma E$, $\gamma \in [2, 4]$

Clustering and ACO-Based Anomaly Detection

Once the data has been gathered and abnormalities discovered, clustering and Ant Colony Optimization (ACO) are used to improve the anomaly identification procedure.

Clustering: Clustering is a technique for grouping data points based on common traits, which may aid in distinguishing between normal and abnormal behavior. K-means and DBSCAN algorithms are used to cluster network traffic data based on traffic rate, packet loss, delay, and energy usage. Anomalous data points that do not fit into any cluster or emerge as outliers in sparse clusters are readily spotted.

Ant Colony Optimization (ACO): ACO is used to improve various important parts of the anomaly detection process, including:

1. **Clustering Parameters:** ACO optimizes the number of clusters for more accurate anomaly detection.

2. **Node Selection for Fog location:** In the smart city network, ACO helps to determine the appropriate location of fog nodes based on network traffic patterns, which improves performance and security.

3. **Path Optimization:** ACO may also optimize data pathways, assisting in the identification of aberrant or malicious network routes, hence improving security and anomaly detection.

Using ACO for parameter optimization and pathfinding assures that the anomaly detection system can handle dynamic and complicated network behaviors.

Step 4 Clustering

1: Input: Data points X = {x1, x2, . . . , xn}, number of clusters k
2: Initialize centroids µ1, . . . , µk
3: repeat
4: Assign each point xi to the nearest cluster using:

$$C_i = \arg\min_j \| x_i - \mu_j \|^2$$

5: Update centroids:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

6: until centroids converge

7: Anomaly Score (distance from centroid):

$$A_i = \| x_i - \mu_{C_i} \|^2$$

8: Mark xi as anomaly if Ai > θ

Step 5 ACO for Optimization

1: Initialize pheromone matrix τij and heuristic matrix ηij
2: for each iteration do
3: for each ant do
4: Construct solution using transition probability:

$$P_{ij} = \frac{T_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \in N_i} T_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}}$$

5: end for
6: Update pheromones:

$$T_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{ants} \Delta\tau_{ij}^{ant}$$

where:

$$\Delta\tau_{ij}^{ant} = \begin{cases} \frac{Q}{L_{ant}} if\ edge(i,j)\ is\ used\ by\ ant \\ 0 \quad otherwise \end{cases}$$

7: end for
8: Output best solution

Simulation and Optimization in OMNeT++

The simulation runs in OMNeT++, which includes both clustering and ACO algorithms. During the simulation, the data created by network operations is recorded in real time. Clustering methods are used to categorize normal activity and identify abnormalities, while ACO is used to optimize network parameters and improve overall anomaly detection efficiency. These new approaches may be evaluated and verified in a realistic network environment, thanks to OMNeT++'s extensive simulation capabilities.

Post-Processing and Evaluation

After the simulation runs and the data is analyzed, the outcomes are assessed using a variety of performance measures. The system's identified anomalies are compared against ground truth labels (normal vs. particular anomaly kinds) to determine the detection system's accuracy. The clustering and ACO-based anomaly detection system's efficacy is assessed using metrics such as Precision, Recall, F1-Score, and ROC AUC. Any false positives or missing abnormalities are reported, and the system is fine-tuned for better performance.

Step 6 Evaluation and Post-Processing

1: Input: True labels Y , Predicted labels Yˆ
2: Compute:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN}$$

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP}$$

$$F1 - Score = \frac{2 \cdot \mathrm{Pr}ecision \cdot \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call}$$

3: Plot ROC curve, compute AUC

5 RESULTS AND DISCUSSION

5.1 Results

In this part, we report the findings from anomaly detection simulations performed in the context of a Smart City Network. The simulation used the OMNeT++ framework to discover anomalies using Ant Colony Optimization (ACO) and Clustering algorithms. These strategies were implemented in the system to identify rogue nodes, traffic surges, and sensor failures. The system's performance was examined in three scenarios: baseline, high sensitivity, and low sensitivity, employing important performance indicators such as accuracy, precision, recall, and F1 score.

	Scenario	Accuracy	Precision	Recall	F1 Score
With anomaly	Baseline	0.924	0.886	0.793	0.837
	High Sensitivity	0.896	0.802	0.943	0.867
	Low Sensitivity	0.917	0.945	0.644	0.767

		Accuracy	Precision	Recall	F1 Score
Without anomaly	Baseline	0.865	0.69	0.598	0.64
	High Sensitivity	0.852	0.672	0.54	0.598
	Low Sensitivity	0.84	0.658	0.471	0.55

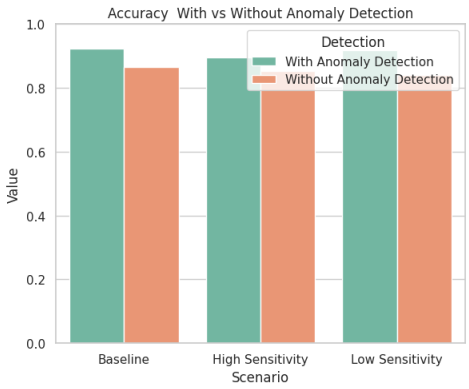


Figure 7 Comparison of Accuracy with and without anomaly detection

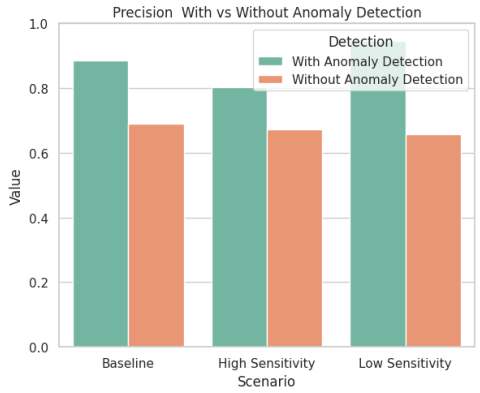


Figure 8 Comparison of Precision with and without anomaly detection

Figure 9 Comparison of Recall with and without anomaly detection

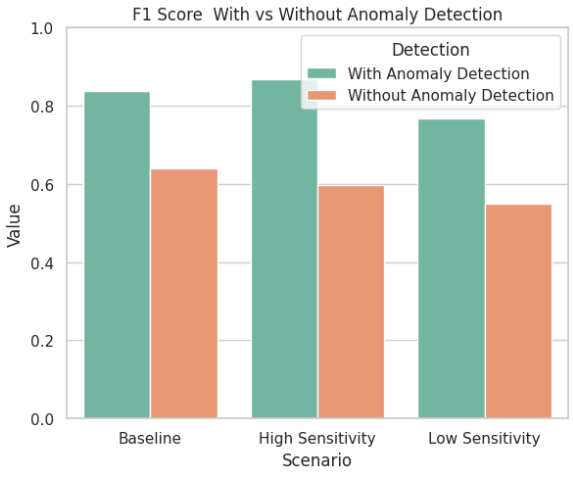
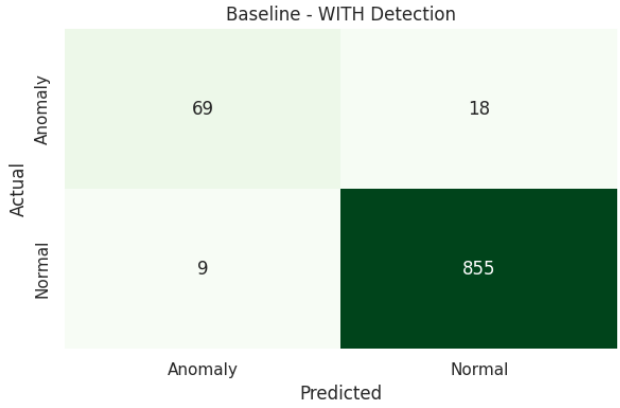


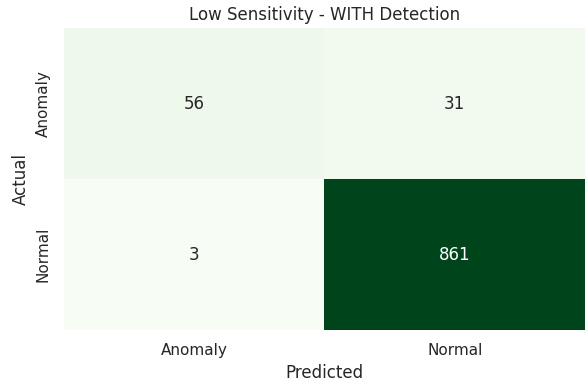
Figure 10 Comparison of F1-Score with and without anomaly detection

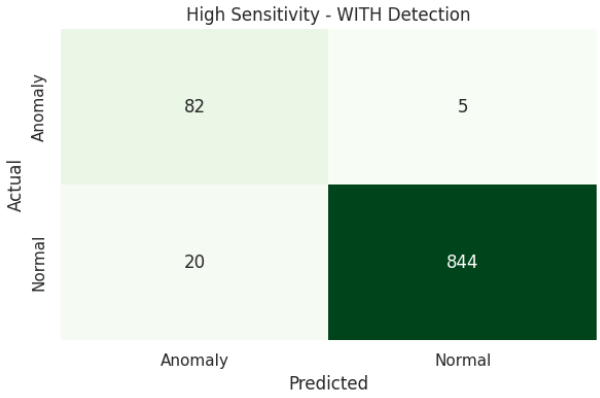
The proposed OMNeT++ + ACO + Clustering-based Framework for Network Anomaly Detection in Smart City Environments performs much better across all evaluation criteria when anomaly detection is used. In the baseline situation, accuracy improved from 0.865 (without detection) to 0.924 (with detection), precision from 0.690 to 0.886, recall from 0.598 to 0.793, and F1 score from 0.640 to 0.837. The High Sensitivity configuration resulted in a considerable increase in recall—from 0.540 to 0.943—as

well as increases in accuracy (0.672 to 0.802) and F1 score (0.598 to 0.867). This trend was also observed in the Low Sensitivity scenario, with accuracy climbing from 0.840 to 0.917, precision from 0.658 to 0.945, recall from 0.471 to 0.644, and F1 score rising from 0.550 to 0.767. These findings clearly show that adding ACO and clustering approaches into the OMNeT++ simulation framework improves network anomaly identification, particularly in dynamic and large-scale smart city scenarios.

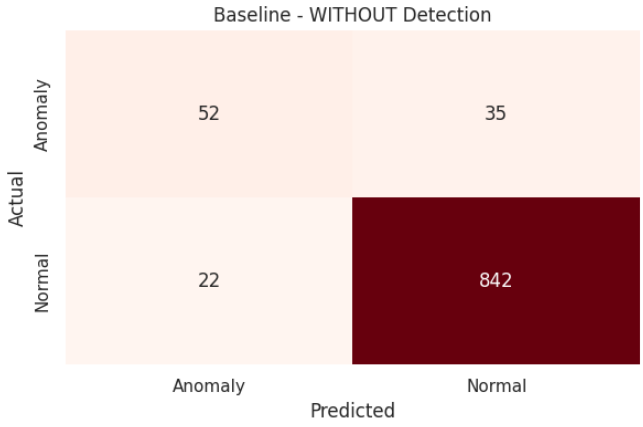
With anomaly detection

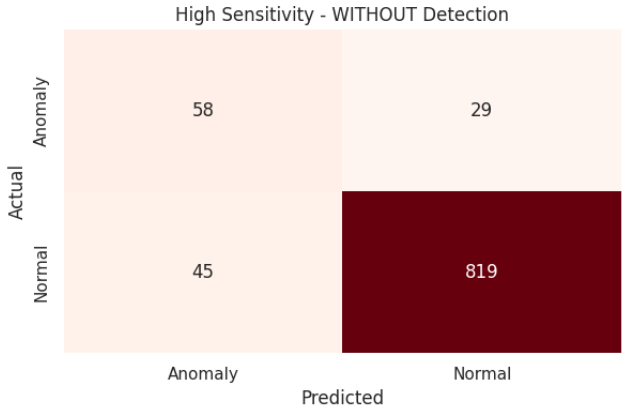


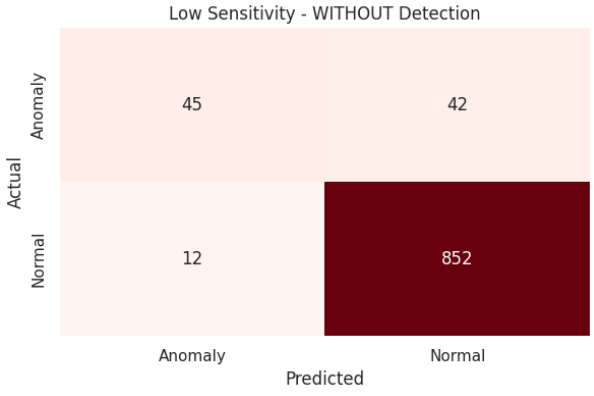




Without anomaly detection







The confusion matrices demonstrate the efficacy of the proposed OMNeT++ + ACO + Clustering-based Anomaly Detection Framework in smart city settings. In the Baseline scenario, with anomaly detection enabled, the model accurately recognized 855 normal and 69 abnormal cases, with just 27 misclassified instances. In comparison, without detection, accurate classifications fell to 842 normal, 52 abnormalities, and 57 misclassifications, indicating a significant decrease in accuracy. In the High Sensitivity scenario, anomaly detection resulted in 82 true positives and 844 true negatives, vs 58 and 819 without it. This indicates an improvement in both anomaly and normal detection accuracy, despite a minor increase in false positives. The Low Sensitivity option further verifies the model's robustness: with detection, just 3 anomalies and 31 normal cases were misclassified, but without detection, misclassifications increased dramatically to 12 anomalies and 42 normal data points. Overall, these comparisons show that adding

anomaly detection considerably decreases false positives and false negatives, increasing the network monitoring system's dependability and accuracy.

Number of Cameras	Fog Latency (ms)	Cloud Latency (ms)	Fog Network Usage (KB)	Cloud Network Usage (KB)	Fog Detection Accuracy (%)	Cloud Detection Accuracy (%)	Fog Processing Efficiency (%)	Cloud Processing Efficiency (%)
1	2.8	4.6	430	580	94.5	88.5	85	65
2	3.5	5.5	880	1,150	94.7	88.8	80	66.5
4	4.5	7.2	1,700	2,300	94.5	87.5	77	61.3
8	5.2	8.5	2,400	3,000	94.2	86.5	75	58
16	6.3	10.1	4,800	5,800	92.9	84.2	73	53.5
32	8	12.2	8,600	10,400	91.4	82.8	70	50.7
64	10	15	14,800	17,600	89.8	80.2	65	45
128	14.5	19	22,300	26,000	87.5	75.5	60	40.5

By incorporating OMNeT++, Ant Colony Optimization (ACO), and clustering into the current system, we can dramatically increase the performance of the smart camera network. ACO improves data routing between fog and cloud nodes, lowering latency at both levels. This reduces fog latency and cloud latency, hence improving real-time processing capabilities. Furthermore, the use of clustering reduces the data load sent from fog nodes to the cloud, resulting in decreased network usage at both the fog and cloud levels. This, in turn, minimizes fog network usage and cloud network usage, both of which are critical for managing big camera networks. The streamlined data flow and edge processing increase detection accuracy at both the fog and cloud levels, with fog detection accuracy benefiting the most from local processing capabilities. While cloud processing efficiency declines marginally when fog nodes handle more data locally, fog processing efficiency rises dramatically owing to improved data flow and decreased redundancy. Overall, these enhancements result in a more efficient and scalable system capable of managing the growing complexity of smart camera networks while maintaining high levels of performance.

6 Conclusion

This research suggested and tested an integrated anomaly detection system for smart city settings that employs OMNeT++, Ant Colony Optimization (ACO), and clustering approaches. The system was meant to identify a variety of network abnormalities, including rogue nodes, sensor problems, and anomalous traffic patterns. The proposed model was evaluated across three sensitivity scenarios—baseline, high, and low—using performance measures such as accuracy, precision, recall, and F1-score. When anomaly detection techniques were used, the results showed substantial performance improvement. For example, in the baseline case, accuracy increased from 86.5% to 92.4%, and the F1-score increased from 0.64 to 0.837, showing a strong detection system capable of effectively recognizing anomalies with few false positives and false negatives.

Furthermore, the system's fog-cloud design improved scalability and efficiency, especially as the number of networked smart cameras grew. By using ACO for optimal routing and clustering for effective data aggregation, the framework reduced both fog and cloud network utilization while improving real-time detection. Fog-based processing was particularly successful, with greater detection accuracy and processing efficiency than centralized cloud-based processing. In conclusion, the combination of OMNeT++, ACO, and clustering provides a highly efficient, scalable, and accurate solution to network anomaly detection, helping to construct intelligent and resilient smart city infrastructures.

References

[1] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, "Internet of things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10474–10498, 2021.

[2] T. Anitha, S. Manimurugan, S. Sridhar, S. Mathupriya, and G. C. P. Latha, "A review on communication protocols of industrial internet of things," in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, IEEE, 2022, pp. 418–423.

[3] M. A. Zaidan *et al.*, "Dense air quality sensor networks: Validation, analysis, and benefits," *IEEE Sens. J.*, vol. 22, no. 23, pp. 23507–23520, 2022.

[4] W. Ullah, F. U. M. Ullah, Z. A. Khan, and S. W. Baik, "Sequential attention mechanism for weakly supervised video anomaly detection," *Expert Syst. Appl.*, vol. 230, p. 120599, 2023.

[5] I. Priyadarshini, P. Mohanty, A. Alkhayyat, R. Sharma, and S. Kumar, "SDN and application layer DDoS attacks detection in IoT devices by attention-based Bi-LSTM-CNN," *Trans. Emerg. Telecommun. Technol.*, vol. 34, no. 11, p. e4758, 2023.

[6] I. Priyadarshini, "Anomaly detection of iot cyberattacks in smart cities using federated learning and split learning," *Big Data Cogn. Comput.*, vol. 8, no. 3, p. 21, 2024.

[7] S. Shukla, S. Thakur, and J. G. Breslin, "Anomaly detection in smart grid network using FC-based blockchain model and linear SVM," in *International Conference on Machine Learning, Optimization, and Data Science*, Springer, 2021, pp. 157–171.

[8] K. Prathapchandran and T. Janani, "A trust aware security mechanism to detect sinkhole attack in RPL-based IoT environment using random forest–RFTRUST," *Comput. Networks*, vol. 198, p. 108413, 2021.

[9] H. Cai, B. Xu, L. Jiang, and A. V Vasilakos, "IoT-based big data storage systems in cloud computing: perspectives and challenges," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 75–87, 2016.

[10] P. Shah, A. K. Jain, T. Mishra, and G. Mathur, "IoT-based big data storage systems in cloud computing," in *Proceedings of Second International Conference on Smart Energy and Communication: ICSEC 2020*, Springer, 2021, pp. 323–333.

[11] H. F. Atlam, A. Alenezi, A. Alharthi, R. J. Walters, and G. B. Wills, "Integration of cloud computing with internet of things: challenges and open issues," in *2017 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, IEEE, 2017, pp. 670–675.

[12] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digit. Commun. Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[13] M. Babar, M. S. Khan, F. Ali, M. Imran, and M. Shoaib, "Cloudlet computing: recent advances, taxonomy, and challenges," *IEEE access*, vol. 9, pp. 29609–29622, 2021.

[14] S. S. Gill *et al.*, "AI for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, p. 100514, 2022.
[15] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, 2017.
[16] M. A. Salih, J. Cosmas, and Y. Zhang, "OpenFlow 1.3 extension for OMNeT++," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, IEEE, 2015, pp. 1632–1637.
[17] M. Ushakova, Y. Ushakov, I. Bolodurina, A. Shukhman, L. Legashev, and D. Parfenov, "Creation of adequate simulation models to analyze performance parameters of a virtual fog computing infrastructure," *Procedia Comput. Sci.*, vol. 186, pp. 603–610, 2021.
[18] Z. Khan and P. H. Ips, "Building Resilient Smart Cities: Sustainability and Inclusiveness," in *Fifth World Congress on Disaster Management: Volume V*, Routledge, 2023, pp. 342–350.
[19] Y. Y. Ghadi *et al.*, "Security risk models against attacks in smart grid using big data and artificial intelligence," *PeerJ Comput. Sci.*, vol. 10, p. e1840, 2024.
[20] T. T. Doan, *Smart City Cyber Resilience: Your Perception Matters*. FriesenPress, 2024.
[21] F. W. Alsaade and M. H. Al-Adhaileh, "Cyber attack detection for self-driving vehicle networks using deep autoencoder algorithms," *Sensors*, vol. 23, no. 8, p. 4086, 2023.
[22] Y. Takiguchi and S. Shiono, "Split Training Method to Generate Data Driven Model for Gas Turbine Engine Anomaly Detection," in *Turbo Expo: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, 2020, p. V005T05A027.
[23] C. Nixon, M. Sedky, and M. Hassan, "Salad: An exploration of split active learning based unsupervised network data stream anomaly detection using autoencoders," *Authorea Prepr.*, 2021.
[24] M. Dragoi, E. Burceanu, E. Haller, A. Manolache, and F. Brad, "Anoshift: A distribution shift benchmark for unsupervised anomaly detection," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 32854–32867, 2022.
[25] H. Zhang, Z. Xiao, J. Gu, and Y. Liu, "A network anomaly detection algorithm based on semi-supervised learning and adaptive multiclass balancing," *J. Supercomput.*, vol. 79, no. 18, pp. 20445–20480, 2023.
[26] Y. Labiod, A. Amara Korba, and N. Ghoualmi, "Fog computing-based intrusion detection architecture to protect iot networks," *Wirel. Pers. Commun.*, vol. 125, no. 1, pp. 231–259, 2022.
[27] C. A. de Souza, C. B. Westphall, R. B. Machado, L. Loffi, C. M. Westphall, and G. A. Geronimo, "Intrusion detection and prevention in fog based IoT environments: A systematic literature review," *Comput. Networks*, vol. 214, p. 109154, 2022.
[28] W. Ben Daoud and S. Mahfoudhi, "SIMAD: Secure Intelligent Method for IoT-Fog Environments Attacks Detection.," *Comput. Mater. Contin.*, vol. 70, no. 2, 2022.
[29] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "A DDoS attack mitigation framework for IoT networks using fog computing," *Procedia Comput. Sci.*, vol. 182, pp. 13–20, 2021.
[30] S. R. Zahra and M. A. Chishti, "Fuzzy logic and fog based secure architecture for internet of things (flfsiot)," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–25, 2023.
[31] A. K. Junejo, N. Komninos, and J. A. McCann, "A secure integrated framework for fog-assisted Internet-of-Things systems," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6840–6852, 2020.
[32] S. R. Zahra and M. A. Chishti, "A generic and lightweight security mechanism for detecting

malicious behavior in the uncertain Internet of Things using fuzzy logic-and fog-based approach," *Neural Comput. Appl.*, vol. 34, no. 9, pp. 6927–6952, 2022.

[33] F. Fama, D. F. S. Santos, and A. Perkusich, "Integrating an iot application middleware with a fog and edge computing simulator," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, IEEE, 2020, pp. 1–6.

[34] K. N. Lawal, T. K. Olaniyi, and R. M. Gibson, "Fog computing infrastructure simulation toolset review for energy estimation, planning and scalability," *Int. J. Sustain. Energy Dev.*, vol. 9, no. 1, pp. 421–426, 2021.

[35] D. Gonçalves, C. Puliafito, E. Mingozzi, O. Rana, L. Bittencourt, and E. Madeira, "Dynamic network slicing in fog computing for mobile users in MobFogSim," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, IEEE, 2020, pp. 237–246.

[36] P. Chapagain, A. Timalsina, M. Bhandari, and R. Chitrakar, "Intrusion detection based on PCA with improved K-means," in *International conference on electrical and electronics engineering*, Springer, 2022, pp. 13–27.

[37] R. Zhang, J. Zhang, Q. Wang, and H. Zhang, "DOIDS: an intrusion detection scheme based on DBSCAN for opportunistic routing in underwater wireless sensor networks," *Sensors*, vol. 23, no. 4, p. 2096, 2023.

[38] J. Chen, X. Qi, L. Chen, F. Chen, and G. Cheng, "Quantum-inspired ant lion optimized hybrid k-means for cluster analysis and intrusion detection," *Knowledge-Based Syst.*, vol. 203, p. 106167, 2020.

[39] C. Zhou and Z. Jiang, "Computer network communication security encryption system based on ant colony optimization algorithm," *Procedia Comput. Sci.*, vol. 228, pp. 38–46, 2023.

[40] H. A. Alterazi *et al.*, "Prevention of cyber security with the internet of things using particle swarm optimization," *Sensors*, vol. 22, no. 16, p. 6117, 2022.

[41] A. C. H. Chow and B. P. Zeigler, "Parallel DEVS: A parallel, hierarchical, modular modeling formalism," in *Proceedings of Winter Simulation Conference*, IEEE, 1994, pp. 716–722.

[42] T. Chamberlain, *Learning OMNeT++*. Packt Publishing, 2013.

[43] W. Drytkiewicz, S. Sroka, V. Handziski, A. Köpke, and H. Karl, "A mobility framework for omnet++," in *3rd International OMNeT++ workshop*, 2003.

[44] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, 1997.

[45] A. Hemmati-Sarapardeh, A. Larestani, M. Nait Amar, and S. Hajirezaie, "Training and optimization algorithms," *Appl. Artif. Intell. Tech. Pet. Ind.*, pp. 51–78, 2020, doi: 10.1016/b978-0-12-818680-0.00003-5.

[46] S. M. K. Heris and H. Khaloozadeh, "Ant colony estimator: an intelligent particle filter based on ACOR," *Eng. Appl. Artif. Intell.*, vol. 28, pp. 78–85, 2014.

[47] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, 2008.

[48] A. Haider, M. Adnan Khan, A. Rehman, M. Rahman, and H. Seok Kim, "A real-time sequential deep extreme learning machine cybersecurity intrusion detection system," *Comput. Mater. Contin.*, vol. 66, no. 2, pp. 1785–1798, 2021.

[49] H. Ahmad, M. Zubair Islam, R. Ali, A. Haider, and H. Kim, "Intelligent stretch optimization in information centric networking-based tactile Internet applications," *Appl. Sci.*, vol. 11, no. 16, p. 7351, 2021.

[50] L. F. da Cruz Nassif and E. R. Hruschka, "Document clustering for forensic analysis: An approach for improving computer inspection," *IEEE Trans. Inf. forensics Secur.*, vol. 8, no. 1, pp. 46–54, 2012.
[51] V. K. Dehariya, S. K. Shrivastava, and R. C. Jain, "Clustering of image data set using k-means and fuzzy k-means algorithms," in *2010 International conference on computational intelligence and communication networks*, IEEE, 2010, pp. 386–391.
[52] K. P. Sinaga and M.-S. Yang, "Unsupervised K-means clustering algorithm," *IEEE access*, vol. 8, pp. 80716–80727, 2020.
[53] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "DBSCAN: Past, present and future," in *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, IEEE, 2014, pp. 232–238.
[54] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data Min. Knowl. Discov.*, vol. 2, pp. 169–194, 1998.
[55] S. Su *et al.*, "N2DLOF: A new local density-based outlier detection approach for scattered data," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2017, pp. 458–465.