A Machine Learning Framework for detecting anomalies in IoT Systems for Smart City Architecture using Autoencoder

Mikin K. Dagli[1*], Dr. Harsha Padheriya[2], Dr. Khyati Rami[3]

[1*]Research Scholar, Department of Computer Engineering, Faculty of Engineering &Technology, Monark University, Ahmedabad, India
mikin.dagli@sal.edu.in
[2] Department of Computer Engineering, Monark University, Ahmedabad, India
hpadheriya@gmail.com
[3] Department of Computer Applications, SAL Institute of Technology and Engineering Research, Ahmedabad, India
khyati.rami@sal.edu.in

Abstract

The rapid expansion of Internet of Things (IoT) devices within smart city infrastructures has created significant challenges in protecting complex and diverse networks from cyberattacks and system anomalies. Conventional detection methods, often rule-based, lack the flexibility to handle the evolving, large-scale nature of IoT data. This study introduces a machine learning-based framework utilizing unsupervised deep learning—specifically, autoencoders for detecting anomalies in smart city IoT environments. The framework is tested on the TON_IoT dataset, which mirrors real-world network traffic and telemetry data from smart ecosystems. By training the autoencoder solely on benign data, the model learns efficient internal representations of normal activity. Deviations from this learned behaviour, identified through high reconstruction errors, signal potential anomalies. Experimental findings reveal that the model achieves strong performance in identifying both cyber threats and operational irregularities, with high precision, recall, F1 Score and overall accuracy. This work underscores the practical benefits of autoencoder-driven detection mechanisms and provides actionable guidance on feature reduction, threshold selection, and deployment scalability for securing IoT-based smart city systems.

1. Introduction:

Numerous networked devices, including smart grids, traffic control systems, environmental sensors, and surveillance units, have been seamlessly integrated into smart cities, which have grown rapidly due to the widespread deployment of Internet of Things (IoT) technologies[4]. Large amounts of real-time data are continuously produced by these devices, allowing for sophisticated features like proactive public

safety systems, efficient energy management, and autonomous transportation. Notwithstanding these benefits, there are serious security and dependability issues with IoT infrastructures due to their interconnectedness and high data throughput. These networks are vulnerable to a range of cyberthreats and operational irregularities because to their vast diversity and size, which could result in service interruptions or sensitive data breaches[24].

Traditional security solutions, such as intrusion detection systems [6], [7] (IDS) that are rule-driven or signature-based, frequently have trouble identifying emerging or changing threats. These techniques' adaptability in intricate and high-dimensional IoT[4] systems is limited by their reliance on preset rules and known threat signatures. Intelligent detection frameworks that can adapt to invisible behaviours and spot small anomalies without the requirement for large labelled datasets are vital as smart cities continue to expand.

Deep learning (DL) and machine learning (ML) approaches [6], [7], [8] have become useful instruments to close this gap. Unsupervised models, such as autoencoders [6], [7], [10], have becoming increasingly popular for anomaly detection. In order to detect aberrant behavior through large reconstruction mistakes when faced with anomalous inputs, autoencoders [6], [7], [10] learn the innate patterns of normal data. Autoencoders are ideal for real-world IoT applications where it is hard to acquire labelled data for every possible threat since they do not require labelled attack data.

In order to identify abnormalities in IoT-driven smart city environments[12], this study proposes a machine learning framework built on an autoencoder-based architecture. The TON_IoT dataset [18], which includes extensive telemetry and network data reflecting typical smart city operations, is used to validate the suggested solution. The autoencoder can learn predicted patterns of system behaviour because it is trained only on benign data samples. When reconstruction[30] mistakes are greater than a predetermined threshold, anomalies, such as cyber attacks and operational disturbances, are reported. According to experimental data, the model successfully and accurately detects a variety of harmful actions with few false alarms.

This framework advances the development of intelligent, robust, and adaptive security systems intended to protect smart city IoT[4] ecosystems by incorporating deep learning techniques into anomaly detection[1].

1.1 Introduction to Anomaly Detection (AD) [1], [6][1]

The practice of locating data instances or patterns that substantially depart from expected behaviour is called anomaly detection (AD)[1], sometimes referred to as outlier detection. To maintain safety, dependability, and security in smart settings (such as smart homes, smart cities, and smart industries), it is essential to identify these anomalies, which may be signs of flows, mistakes[9].

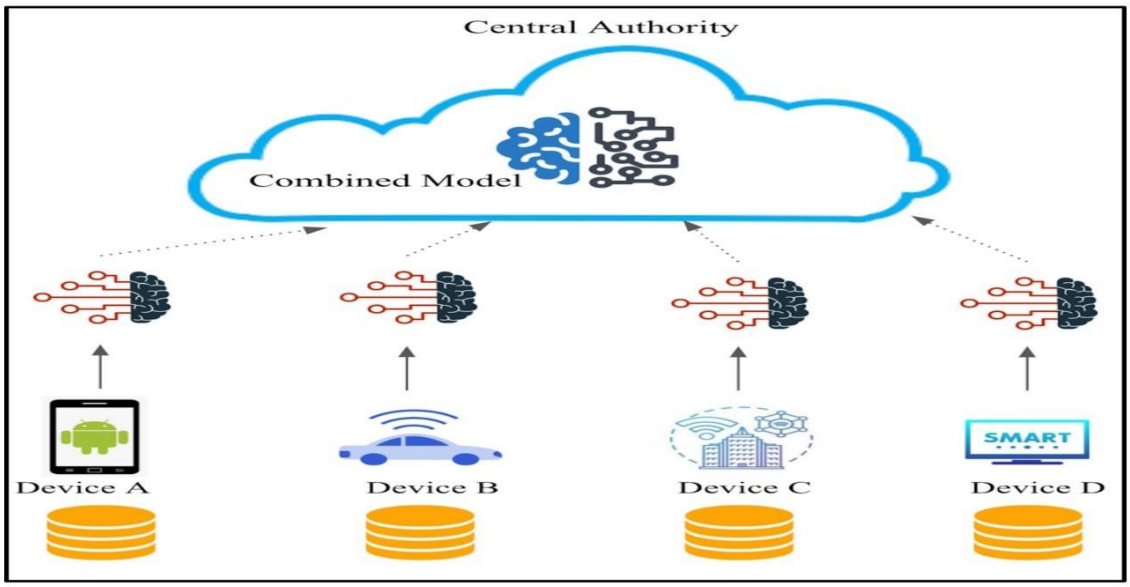

Figure 1:Anomaly Detection of IoT Cyber attacks in Smart Cities

Multivariate time-series data streams are produced in smart environments by the networked IoT devices[4]. These data streams are continuously monitored by AD algorithms to spot anomalous activity that can jeopardize operations or safety, like device malfunctions, security lapses, or system breakdowns.

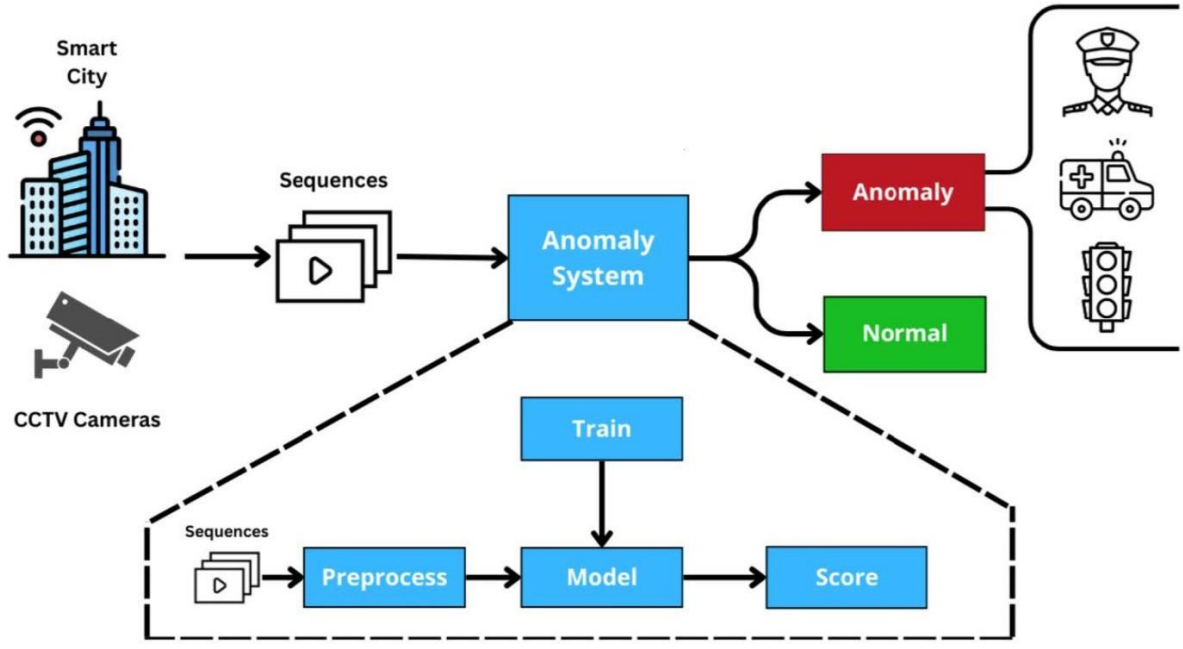

Figure 2:Common Architecture of anomaly detection systems in traffic surveillance

1.1.1 Types of Anomalies

1. **Point Anomalies**:
    - A single data instance that is significantly different from the rest.
    - Example: A temperature sensor suddenly reads 100°C while others are around 25°C.
2. **Contextual Anomalies**:
    - A data instance that is only anomalous within a specific context (time, location, etc.).
    - Example: 30°C is normal in summer but anomalous in winter.
3. **Collective Anomalies**:
    - A group of data points is anomalous together, though individual points may appear normal.
    - Example: A sudden group of traffic events indicating a blockage or accident.

1.1.2 Application Scenarios

AD plays a vital role across various smart environments:

- **Smart Homes:** Detecting fire hazards, energy overuse, or unusual activity patterns.
- **Smart Transport:** Monitoring traffic congestion, identifying accidents, and abnormal vehicle behaviour.
- **Smart Industry:** Detecting equipment failure, cyber-attacks, or production anomalies.
- **Smart Health / Wearable:** Identifying unusual physiological signals like heart rate or motion.

These applications often depend on IoT-generated **time-series data**, requiring real-time and context-aware detection.

1.1.3 Development Challenges in Smart Environments

1. **Scarcity of Labelled Data:**
    - Anomalies are rare; labelling is time-consuming and expensive.
    - Leads to a need for **unsupervised or semi-supervised** methods.
2. **Data Complexity:**

- IoT data is high-dimensional, multi-modal, and often non-stationary.
- Includes spatial and temporal dependencies.

3. **Integration of Heterogeneous Data:**
   - Combining structured (sensor) and unstructured (e.g., complaints) data is challenging.
4. **Granularity Issues:**
   - Finding the right level of spatial/temporal resolution to detect relevant anomalies.
5. **Contextualization Requirements:**
   - AD must consider time, location, and relational context.
   - Example: Normal activity at 2 PM could be anomalous at 2 AM.
6. **Privacy and User Acceptance:**
   - Personal sensor data raises concerns.
   - False positives/negatives, lack of explain ability, and intrusiveness reduce trust

2. Literature Survey :

2. 1. Anomaly Detection in Smart Environments A Comprehensive Survey

In order to automate processes and improve user experiences, smart environments—from homes and cities to industrial systems—rely on networked sensors and gadgets. These settings are more susceptible to abnormalities, though, which could be signs of intrusions, device failures, or strange activity. The complexity, amount, and variety of data in smart settings are frequently too much for traditional rule-based anomaly detection systems to handle. Consequently, machine learning (ML) has become a potent and flexible method for effectively and precisely identifying anomalies in these kinds of systems[1].

Objective:

This survey's primary goal is to present a thorough overview of the machine learning methods currently in use for anomaly[1] detection in smart environments. It seeks to investigate the different machine learning models used, assess their efficacy, and pinpoint the main implementation obstacles. The study also aims to serve as a resource for both scholars and practitioners by highlighting developments, trends, and future directions in this field.

Scope:

Smart homes, smart healthcare, smart grids, and smart transport systems are just a few of the many applications in smart environments that are included in this survey. It examines supervised, unsupervised, and semi-supervised learning techniques for anomaly detection[1] in real time and offline. Additionally, it takes into account how the selection of anomaly detection methods is influenced by system needs including data heterogeneity, scalability, real-time processing, energy efficiency, and privacy issues.

Problem Statement:

In smart environments, anomaly detection[1] poses a number of significant difficulties. First of all, the generated data is frequently high-dimensional, noisy, and heterogeneous, making it challenging to analyse with conventional techniques. Second, a lot of machine learning models need a lot of labelled data, which is difficult to get in practical situations. Third, it can be difficult to discern between infrequent but valid events and real abnormalities, which can result in significant false positive rates. Furthermore, real-time detection with low latency and processing overhead is frequently needed in smart environments, which is challenging to accomplish with intricate models. The use of anomaly detection solutions in sensitive contexts is made more difficult by security and privacy concerns.

Overcoming the Challenges:

Researchers have used a number of creative approaches to overcome these problems. Because they don't require as much labelled data, unsupervised and semi-supervised learning techniques are more useful in practical settings. Complex patterns and temporal connections in data can be captured by deep learning models like autoencoders [6], [7], [10] and recurrent neural networks (like LSTM). By training models across decentralised data sources without exchanging raw data, federated[8] learning provides privacy-preserving solutions. By enabling anomaly detection directly on devices, edge computing[3] lowers bandwidth consumption and latency. Lastly, hybrid models can enhance interpretability and detection accuracy by fusing machine learning with statistical or rule-based methods.

2.2 Anomaly detection using autoencoders [6], [7], [10] with network analysis features

Introduction:

Finding anomalous patterns in complicated systems is crucial, particularly when it comes to network security and monitoring. The capacity of conventional statistical methods to represent high-dimensional, non-linear data is frequently constrained. By learning compact representations of data and detecting abnormalities based on reconstruction[[30] errors, autoencoders [6], [7], [10]—a kind of deep learning model—offer a potent remedy. Autoencoders can improve the detection of subtle, context-aware anomalies in network behaviour when paired with network analysis parameters including traffic flow, connection patterns, and node centrality.

Objective:

This study aims to investigate the efficacy of autoencoders [6], [7], [10] for anomaly identification when enhanced with network analysis-derived features. Evaluating how these integrated methods may increase detection accuracy, lower false positives, and identify intricate temporal and structural patterns in network data is the aim.

Scope:

With a focus on incorporating network-based metrics including degree distribution, clustering coefficients, and community recognition, this work applies deep autoencoder models to identify anomalies in network data. Numerous use-cases are covered by the analysis, such as performance monitoring in distributed systems, identifying cybersecurity threats, and identifying unusual user behaviour on social networks.

Problem Statement:

Although autoencoders [6], [7], [10] are good at identifying patterns in high-dimensional data, they could have trouble comprehending context when used only with network-based data. Although network analysis features offer useful information, it can be challenging to successfully incorporate them into deep learning models. Pre-processing graph-based features, striking a balance between model complexity and real-time performance requirements, and making sure that models continue to be generalisable across various network topologies and scales are some of the primary problems.

Overcoming the Challenges:

A number of tactics can be used to address these issues. Before supplying network topologies to autoencoders [6], [7], [10], useful characteristics can be extracted using graph embedding techniques like node2vec or GCNs. Model efficiency is maintained

through dimensionality reduction and normalisation. Contextual knowledge can be enhanced by hybrid architectures that integrate autoencoders [6], [7], [10] with attention mechanisms or graph neural networks. To lessen reliance on labelled datasets, unsupervised and semi-supervised learning techniques might be applied.

2.3 Unsupervised Anomaly Detection in IoT system for smart cities

Introduction:

IoT (Internet of Things)[4] devices have become essential for automation, monitoring, and service delivery as smart cities have grown in popularity. Large volumes of data are gathered in real-time by these networked systems, facilitating public safety services, energy optimisation, traffic control, and smart infrastructure. However, risks and unexpected behaviour are introduced by the distributed and dynamic nature of IoT networks. IoT[4] data anomalies could be a sign of malfunctions, security risks, or problems with operations. In this setting, unsupervised anomaly detection is especially useful since it can find anomalous patterns without the need for tagged data, which is frequently inaccessible or limited in large-scale urban systems.

Objective:

Investigating and assessing unsupervised learning techniques for anomaly detection in IoT systems in smart cities is the aim of this project. In order to improve the resilience and dependability of smart city infrastructure, methods for precisely identifying variations in sensor readings, network traffic, and system behaviour are sought after[3].

Scope:

This study covers a number of topics related to smart cities, such as energy grids, smart buildings, public safety networks, intelligent transportation systems, and environmental monitoring. The emphasis is on deep learning models like autoencoders [6], [7], [10] and isolated forests, as well as unsupervised methods like clustering and dimensionality reduction, which can identify anomalies as departures from learnt patterns after learning normal patterns from unlabelled data.

Problem Statement:

There are various obstacles to anomaly detection when IoT[4] is implemented in smart cities. High volume, high velocity, and frequently noisy or incomplete data are produced. It is not feasible to label such data for supervised learning. Furthermore, anomalies can take many other forms, such as equipment breakdowns, cyberattacks, or odd user

behaviour. One of the biggest challenges is still ensuring detection accuracy without tagged data while preserving scalability and real-time performance[5].

Overcoming the Challenges:

To tackle these challenges, unsupervised learning approaches leverage statistical analysis, clustering algorithms (e.g., DBSCAN, k-means), and advanced models like variation autoencoders[10] and GANs (Generative Adversarial Networks). Feature engineering based on temporal and spatial context can enhance anomaly representation. Dimensionality reduction techniques such as PCA and t-SNE help manage complexity. Edge computing and fog computing paradigms support real-time detection by processing data closer to the source. Additionally, ensemble methods and adaptive models improve robustness and adaptability in dynamic urban environments.

2.4 Comparisons of Literature Paper

Table 1:Literature Paper comparisons

Criteria	**Paper 1: Comprehensive Survey**	**Paper 2: Autoencoders + Network Analysis**	**Paper 3: Unsupervised IoT Anomaly Detection**
Focus	Broad survey of anomaly detection in smart environments	Deep learning (Autoencoders) for network anomaly detection	Unsupervised learning for IoT in smart cities
Methodology	Survey of statistical, ML, DL, hybrid techniques	Autoencoders + graph/network metrics	Clustering, dimensionality reduction (e.g., PCA, t-SNE)
Environment	General smart environments (homes, buildings, etc.)	Computer networks (possibly smart environments)	IoT in smart cities
Learning Type	Supervised, unsupervised,semi-supervised	Unsupervised	Unsupervised
Novelty	Integration and	Combination of	Focused approach

Criteria	**Paper 1: Comprehensive Survey**	**Paper 2: Autoencoders + Network Analysis**	**Paper 3: Unsupervised IoT Anomaly Detection**
	comparison of multiple methods	network topology and deep learning	for smart cities with real IoT data
Evaluation	Comparative analysis from literature	Custom dataset + network simulations	Real-world IoT data, performance benchmarking
Strengths	Comprehensive overview; good for beginners	Accurate detection using structure + features	Adaptable to new environments, no need for labeled data
Weaknesses	No new technique proposed	Might not generalize to non-network anomalies	Accuracy depends on good feature extraction
Use Cases	Broad: smart homes, industry, health, etc.	Network intrusion, security anomalies	Traffic, environment, utility monitoring
Techniques Covered	KNN, SVM, PCA, Isolation Forest, DL	Autoencoders, Graph Analysis, Feature Engineering	DBSCAN, PCA, Isolation Forest

Aspect	**Survey (P1)**	**Autoencoder + Net (P2)**	**IoT Unsupervised (P3)**
Scope	Broad	Focused	Focused
Learning Approach	Mixed	Unsupervised	Unsupervised
Domain	Smart Environments	Network Systems	Smart Cities (IoT)
Innovation Level	Medium	High	Medium-High
Real-world Evaluation	Low	Medium	High
Scalability	High	Medium	High

Aspect	**Survey (P1)**	**Autoencoder + Net (P2)**	**IoT Unsupervised (P3)**
Complexity	Medium	High	Medium

3. Related Work:

Detecting unusual or abnormal behaviour, also known as anomaly detection, in IoT systems and smart cities has been extensively researched because it plays a crucial role in ensuring security and smooth operation[2]. Various methods have been explored, including statistical models, clustering techniques [26], [27], and supervised machine learning [6], [7].

Statistical Methods:

These methods use presumptions about probability distributions or mathematical patterns to analyse data. Data points are marked as anomalies when they substantially depart from these anticipated trends. Nevertheless, these techniques frequently presume that data adheres to particular distributions, which isn't necessarily the case in intricate IoT settings[11]. They may also have trouble handling datasets with a lot of features (high-dimensional data) and misinterpret regular but unexpected data as anomalies.

Clustering Techniques:

Data is grouped using clustering into groups of related points. Anomalies are data points that are distant from clusters or don't fit well into any cluster. Despite its benefits, clustering necessitates careful adjustment of certain factors, such the number of clusters[16]. Additionally, it loses some of its effectiveness when dealing with high-dimensional data[32] and when the presumption that normal data forms distinct clusters is broken.

Supervised Machine Learning:

Labelled datasets with examples of normal and anomalous data are used by supervised techniques such as Support Vector Machines (SVM), Random Forests, and Neural Networks. They use this training to learn how to categorise new data. However, due to the rarity and diversity of anomalies, labelled anomaly data is difficult to gather[1].

Models that have been trained on known anomalies might not be able to identify novel or distinct anomalies.

Why Autoencoders?

Autoencoders, a type of neural network, provide a powerful alternative because they do not require labelled data and can learn complex data patterns. They rebuild the data after compressing it into a smaller representation. While abnormalities lead to higher reconstruction errors, normal data can be accurately recovered. This characteristic facilitates the efficient detection of anomalies. Autoencoders are ideal for anomaly detection in smart city architectures because they can also handle high-dimensional data effectively and adjust to changing trends in IoT systems[1].

3.1 Objective and Scope:

To develop an unsupervised anomaly detection method for IoT systems in smart cities, aiming to improve the reliability, security, and robustness of these systems without the need for labelled datasets[1].

The objective of this study is to develop and evaluate a machine learning-based anomaly detection framework that utilizes autoencoder-based deep learning models to identify abnormal behaviour in IoT systems within smart city architectures. The aim is to enhance the security, reliability, and situational awareness of smart urban environments by detecting cyber-attacks, device malfunctions, and unexpected operational patterns in real-time.

Scope:

1. **Smart City IoT Context[18]:** The framework focuses on detecting anomalies in critical IoT applications used in smart cities such as traffic monitoring, environmental sensing, energy metering, and public safety systems.
2. **Use of Autoencoder Algorithm[10]:** The study employs unsupervised deep learning (autoencoders [6], [7], [10]) to learn the normal behaviour of multivariate IoT data and detect deviations (anomalies) without requiring labelled anomaly data.
3. **Data Sources:** The framework is tested and validated using publicly available smart city-related datasets (e.g., BoT-IoT, TON_IoT, UNSW-NB15) that represent real-world IoT network traffic and telemetry.

4. **Performance Metrics[2]:** Evaluation is based on standard metrics such as precision, recall, F1-score, ROC-AUC, and reconstruction error to measure the effectiveness and reliability of the anomaly detection model.
5. **System Characteristics:** The framework is designed to be scalable, real-time capable, and suitable for deployment in resource-constrained IoT environments typical of smart city systems.
6. **Security and Operational Integrity:** The solution aims to support early detection of network threats, sensor faults, and unauthorized behaviours, contributing to resilient smart city infrastructure[4].

3. 2 Key Contributions:

- **No Need for Labelled Data:** Uses unsupervised methods, which are suitable for dynamic and evolving IoT environments.
- **Scalable Architecture:** Designed for large-scale deployment across various smart city infrastructures.
- **Generic Framework:** Can be adapted to different types of sensors and urban applications.

3.3 Research Gap

Despite significant advancements in anomaly detection using machine learning and deep learning, several critical gaps remain when applying these techniques to IoT systems within smart city architectures, particularly when using autoencoders [6], [7], [10]:

1. **Lack of Domain-Specific Models for Smart Cities[12]:** Most existing autoencoder-based anomaly detection models are generic and not tailored to the heterogeneous nature of smart city IoT data, which includes environmental sensors, traffic flows, surveillance feeds, and utility meters. These systems have unique patterns, dynamics, and risk profiles that are not adequately captured by general models.
2. **Insufficient Handling of Contextual and Collective Anomalies[15]:** Autoencoders primarily excel at detecting point anomalies but often fail to capture contextual anomalies (e.g., normal in one time/location, abnormal in another) and collective anomalies (e.g., a group of otherwise normal data points forming an unusual pattern), which are common in time-series IoT data from smart cities.

3. **Limited Unsupervised Approaches with High Accuracy[16]:** While autoencoders [6], [7], [10] are naturally unsupervised, many models either assume access to some labelled data or suffer from high false positive rates due to noise and drift in real-world IoT environments. There is a need for robust unsupervised approaches that can learn accurate normal behaviour in diverse and evolving settings.
4. **Scalability and Resource Constraints Not Fully Addressed:** Many deep learning models require high computational resources, making them impractical for real-time or edge deployment in resource-constrained IoT devices. There is a gap in designing lightweight autoencoder architectures that are optimized for on-device or distributed processing[3].
5. **Inadequate Benchmarking on Realistic Datasets[27]:** A significant number of studies use outdated or overly synthetic datasets. There is a lack of comprehensive evaluation on recent, diverse, and realistic IoT anomaly datasets (e.g., TON_IoT, BoT-IoT) that reflect the complexity of smart city environments[1].
6. **Interpretability and Trust:** Autoencoders function as black-box models, making it difficult for city operators to understand why an anomaly was detected. There is a research gap in developing explainable deep learning-based[6] AD models to improve trust and usability in smart city operations[1].

4. Methodology

4.1 Framework Overview (Reworded Explanation)

A number of crucial components make up the suggested machine learning framework for identifying irregularities in IoT systems for smart cities. Together, these components process the unprocessed sensor data, determine what constitutes "normal," and identify any anomalous behaviour without the requirement for labelled samples[3].

1. Data Preparation

IoT data can be messy — sensors might give faulty readings, or some data points could be missing. So, the first step is to clean and prepare this data:

- **Removing Noise:** Errors and sporadic fluctuations that don't accurately reflect actual conditions are eliminated.

- **Scaling Data:** Errors and intermittent variations that are not representative of the real situation are removed.
- **Handling Missing Data:** Sometimes transmission issues or sensor failure result in missing data. To keep the data complete, we fill in these gaps with techniques like interpolation and averaging numbers that are close together.

2. Feature Extraction

IoT sensors collect a lot of data over time, often with many different measurements. To make it easier to analyze:

- **Summarizing Time Windows:** We compute statistics like averages or maximum values to summarize data across brief intervals (such as every few minutes) rather than examining each individual data point.
- **Reducing Dimensions:** In order to assist the model concentrate on what really matters, we use techniques like Principal Component Analysis (PCA) to decrease the quantity of features while retaining the crucial information.

3. Autoencoder Model

The core of the framework is an autoencoder, a type of neural network that learns to represent the normal data efficiently:

- **How it Works:** After reducing the size of the input data, the autoencoder attempts to restore it to its original form.
- **Training:** Since it has only been trained on typical data, it picks up typical patterns and characteristics without noticing any irregularities.
- **Unsupervised:** This approach is highly useful in situations where it is difficult to locate such labelled data because it does not require examples of anomalies.

4. Detecting Anomalies

Once trained, the autoencoder is used to check new data:

- **Calculating Error:** It computes the difference (referred to as reconstruction error) between the fresh data and its reconstruction.
- **Flagging Anomalies:** The data point is identified as an anomaly if the difference exceeds a predetermined threshold. The error distribution observed during training can be used to dynamically set the threshold.

5. Flexibility and Scalability

- The framework can be updated regularly with new data to adapt to changing conditions in the smart city.

- It is scalable for big systems like energy grids, environmental sensors[3], and traffic monitoring since it is made to manage data from numerous IoT devices across various applications.

With this method, the system can accurately identify unexpected behaviour without the requirement for labelled samples of anomalies by learning what "normal" means in a complicated IoT environment. It is ideal for practical smart city applications and strikes a balance between precision and adaptability.

4.2.Autoencoder Architecture

The design of the autoencoder is important for learning useful data features and spotting unusual patterns. Here's a detailed look at each part of the architecture:

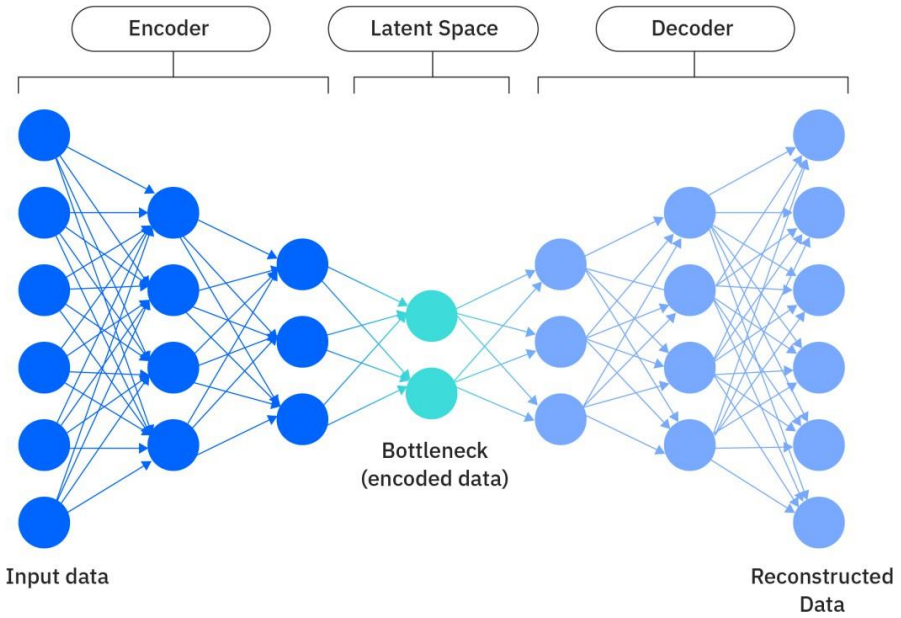

Figure 3: Autoencoder Architecture

Input Layer

- One hundred features, representing various readings from sensors or systems in a smart city, are accepted by the input layer[4]. These features may include traffic flow, temperature, humidity, pollution levels, and other operational and environmental characteristics.
- This layer merely provides the network with the basic data.

Encoder

- By capturing the most crucial information and disregarding less important aspects, the encoder's function is to compress the input.

- Its three layers progressively reduce the data's size:
    - The first layer's 100 neurons correspond to the size of the input.
    - This is reduced to 64 neurons in the second layer.
    - It is further compressed to 32 neurons in the third layer.
- The ReLU activation function is used in each layer, adding non-linearity to aid in the network's learning of intricate patterns.
- This step forces the model to find meaningful, compact representations of the data.
-

Latent Layer

- With 16 neurons—much fewer than the initial 100 features—the latent layer is the smallest and most condensed representation of the input, capturing the essential elements of the input data[4].
- The network is urged to concentrate on the most significant signals in the data due to its limited size.

Decoder

- The decoder reconstructs the original input from the compressed data.
- It mirrors the encoder by expanding the data back:
    - Starts with 32 neurons.
    - Then 64 neurons.
    - Finally, back to 100 neurons to match the original input size.
- Similar to the encoder, it uses nonlinear activations to help recreate the input details.
- The goal is to produce an output close to the original input.

Output Layer

- The reconstructed data is the output, and it should be as similar to the original input as feasible.
- To make the model better, the difference between the original input and this reconstruction is measured during training.
- During training, the difference between the original input and this reconstruction is measured in order to improve the model.
- In contrast to the training data, a substantial reconstruction error for a fresh input suggests that the input is abnormal or unexpected.

This configuration enables the autoencoder to effectively identify abnormalities when they arise and learn the typical behaviour of complicated, high-dimensional data from IoT sensors[3].

5. Experiment Setup

5.1 Dataset: We used the **TON_IoT Dataset (by UNSW Canberra)** dataset and **BoT-IoT** Dataset comprising normal and attack traffic from a smart home network. The dataset includes 345292 records with 43 features.

An autoencoder is a special type of neural network used in machine learning. It helps computers learn to understand and simplify data without needing labels. The main idea is to take input data, shrink it to a smaller form, and then try to rebuild the original data from that smaller version.

An autoencoder has two main parts: the encoder and the decoder.

The encoder takes the input data and compresses it into a smaller, simpler form called a code or latent representation. This helps the system focus only on the most important information in the data.

The decoder takes this smaller code and tries to recreate the original input data. It works in the opposite way of the encoder by expanding the smaller version back to the full size.

The goal of an autoencoder is to make the output as close as possible to the original input. The difference between the input and output is measured using a loss function like mean squared error (MSE) or binary cross-entropy[6].

Autoencoders are useful in many ways:

- **Dimensionality reduction:** They simplify complex data while keeping important features.
- **Anomaly detection:** They help find unusual data that doesn't fit well with what they've learned.
- **Image denoising:** They can clean up noisy images.
- **Data generation:** Advanced types like Variation Autoencoders (VAEs) can create new, similar data[30].

In short, autoencoders [6], [7], [10] are helpful tools for making sense of large, complex data without needing labeled examples. They are used in image processing, data compression, and detecting unusual patterns in data.

The basic structure of an autoencoder looks like this:

Input (X) → Encoder → Latent Representation (Z) → Decoder → Output (X')

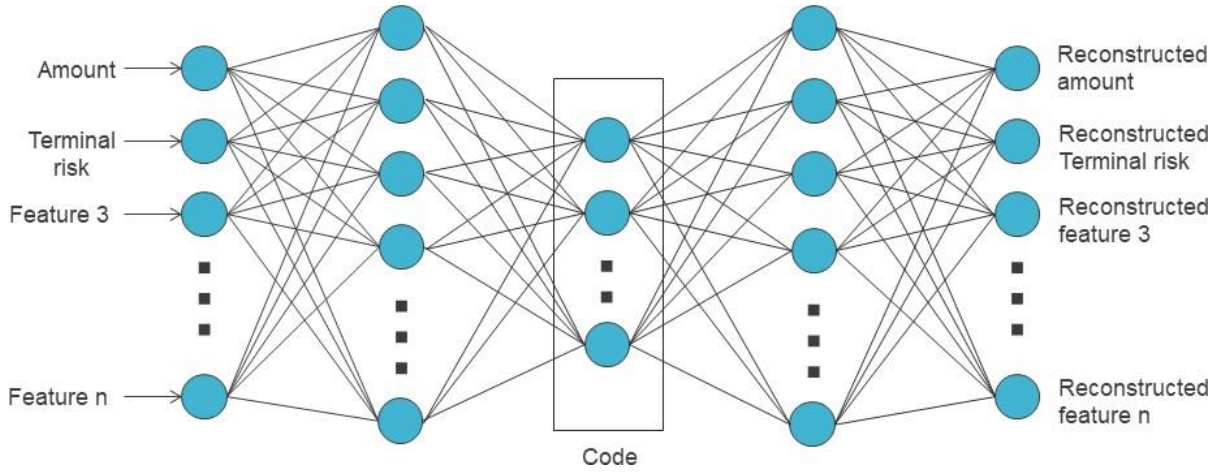

Figure 4:Autoencoders and anomaly detection

5.1.1 TON_IoT Dataset (by UNSW Canberra)

The TON_IoT dataset [18] is a modern cybersecurity dataset developed by the IoT Lab at UNSW Canberra. It is designed to support machine learning and artificial intelligence research in the fields of Internet of Things (IoT) and Industrial IoT (IIoT) security[11].

This dataset was collected from a realistic testbed that included various smart devices, operating systems like Windows 7/10 and Ubuntu 14/18, as well as cloud and edge systems[3]. It combines data from multiple sources, such as:

- Telemetry data from IoT/IIoT sensors (e.g., smart meters, weather sensors),
- Operating system logs (e.g., processes, memory usage, disk activity),
- and network traffic data, including packet and flow-level information.

The TON_IoT dataset [18] includes both normal activities and a wide range of cyber-attacks, making it highly useful for training and testing intrusion detection systems (IDS). It supports both binary classification (normal vs. attack) and multi-class classification (types of attacks).

One of the strongest features of TON_IoT is its realism and scale—it contains millions of records captured across multiple layers such as edge, fog, and cloud. It reflects real-world conditions and includes advanced networking setups like software-defined networking (SDN) and network function virtualization (NFV).

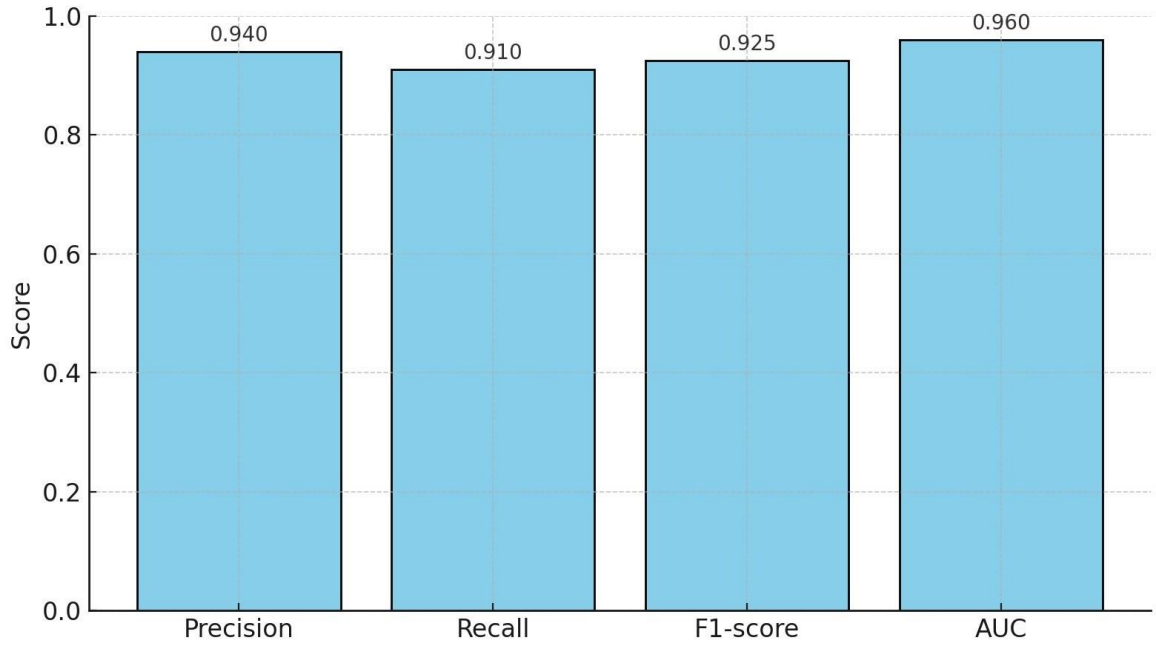

Figure 5:Performance evaluation metrics [2], [26], [27] on TON_IoT Dataset

5.1.2 BoT-IoT Dataset

The BoT-IoT dataset [18] is a large and realistic dataset created by the Cyber Range Lab at UNSW Canberra. It is designed to help researchers build and test machine learning models for detecting cyber-attacks in Internet of Things (IoT) environments.

To build this dataset, the researchers created a smart network environment that included both normal network traffic and a wide variety of simulated cyber-attacks. These attacks included:

- Denial of Service (DoS) and Distributed Denial of Service (DDoS),
- Scanning and reconnaissance (like fingerprinting and port scanning),
- Key logging (capturing keyboard input),
- and data theft or infiltration.

All the network activity was recorded and saved in the form of packet capture (PCAP) files, totalling over 69 GB of data. They also processed this raw data into flow records, which are easier to work with for machine learning. These flow records are available in CSV format and contain around 72 million entries. For quick experiments, a smaller 5% sample of the dataset is also available, which has about 3 million records[2].

Each record in the dataset includes detailed information about the network flow, such as:

- Source and destination IP addresses and ports,

- Protocols used (like TCP, UDP, or HTTP),
- Number of packets, bytes, duration,
- And a label indicating whether the activity is normal or an attack.

The dataset includes a total of 43 features per record, including both standard network metrics and additional statistical values. These features help machine learning models understand patterns and detect unusual or malicious behaviour.

The BoT-IoT dataset [18] is useful for both binary classification (normal vs. attack) and multi-class classification (different types of attacks). It is widely used by researchers to develop and test cyber security models, improve detection accuracy, and deal with challenges like imbalanced data.

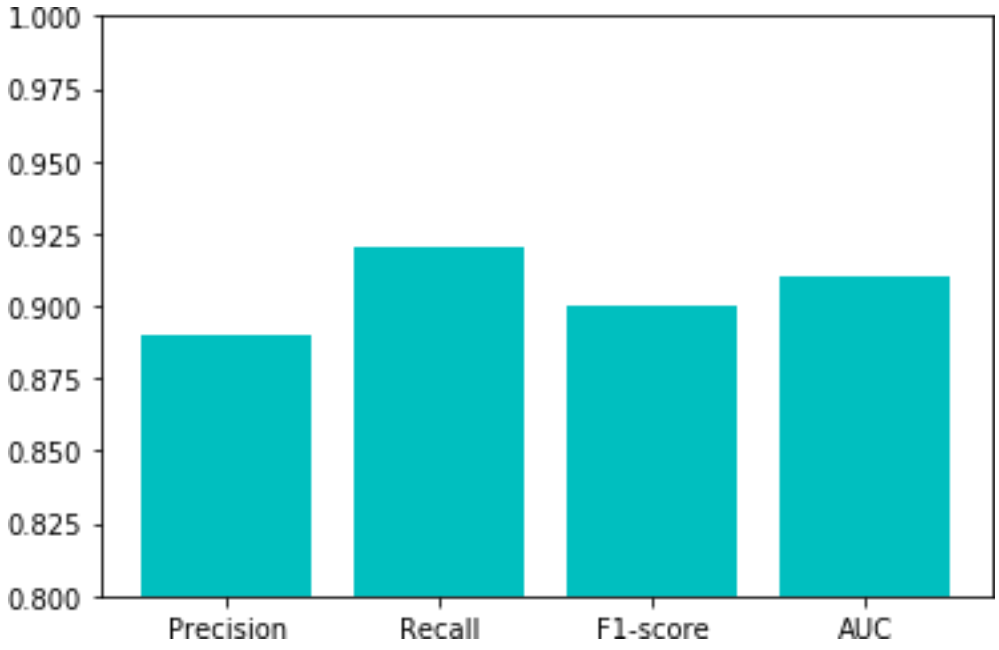

Figure 6: Performance evaluation metrics [2], [26], [27] on BoT_IoT Dataset

5.2 Evaluation Metrics

Evaluation Metrics

5.2.1. Precision:

Precision measures the accuracy of the positive predictions made by the model.

Formula: Precision = TP / (TP + FP)

Interpretation: It indicates how many of the instances predicted as anomalies by the

model are actually true anomalies.

5.2.2. Recall:

Recall measures the ability of the model to correctly identify all positive instances
(anomalies) in the dataset.

Formula: Recall = TP / (TP + FN)

Interpretation: It indicates how many of the true anomalies in the dataset were identified correctly by the model.

5.2.3. F1-score:

F1-score is the harmonic mean of precision and recall, providing a single metric that
balances between them.

Formula: F1-score = 2 * (Precision * Recall) / (Precision + Recall)

Interpretation: It gives a balanced measure of the model's accuracy in detecting anomalies, considering both precision and recall.

5.2.4. Area Under ROC Curve (AUC):

AUC measures the ability of the model to distinguish between positive and negative
classes across various thresholds.

Interpretation: A higher AUC value (closer to 1) indicates that the model is better at
distinguishing between normal and anomalous instances.

5.3 Training Procedure (Detailed Explanation)

The training of the autoencoder involves optimizing the model to accurately reconstruct normal IoT data while minimizing reconstruction errors. The key components of the training process are described below:

5.3.1. Loss Function: Mean Squared Error (MSE) [10], [27]

The Mean Squared Error (MSE) [10], [27] is used as the loss function during training. MSE measures the average squared difference between the input data and its reconstruction by the autoencoder. It is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

- Using MSE encourages the model to minimize the difference between the input and reconstructed output.
- A lower MSE indicates better reconstruction and thus better learning of normal patterns.

5.3.2. Optimizer: Adam

Adam (Adaptive Moment Estimation) is chosen as the optimizer to update the model weights during training.

- Adam combines the benefits of two popular optimization algorithms, AdaGrad and RMSProp.
- It adapts the learning rate for each parameter individually, which speeds up convergence and improves training stability.
- This makes Adam well-suited for training deep neural networks like autoencoders [6], [7], [10].

5.3.3. Epochs: 100

- The model is trained over 100 epochs.
- An epoch is one full pass through the entire training dataset.
- Multiple epochs allow the model to iteratively learn and refine its parameters to reduce reconstruction error.
- The number 100 is a typical choice balancing sufficient learning time without excessive overfitting.

5.3.4. Batch Size: 128

- The training data is divided into batches of 128 samples.
- Instead of processing the entire dataset at once, the model updates weights after each batch.
- Batch training reduces memory consumption and allows for more frequent updates, improving learning efficiency.
- A batch size of 128 is a common compromise between stable gradient estimation and computational speed.

5.3.5. Early Stopping: Monitored Validation Loss

- Early stopping is used to prevent overfitting and improve generalization.
- During training, the model's performance on a separate validation set is monitored.

- If the validation loss (reconstruction error on unseen data) stops improving for several consecutive epochs, training is halted early.
- This technique saves time and avoids the model learning noise or irrelevant details in the training data.

By using MSE as the loss, the Adam optimizer [10] for efficient weight updates, a suitable number of epochs and batch size, and early stopping based on validation loss, the training procedure ensures the autoencoder learns to reconstruct normal IoT data accurately. This trained model can then effectively identify anomalies by detecting when reconstruction errors exceed a set threshold.

6. Results and Discussion

Table 2: Performance Comparison

Metric	Autoencoder	Isolation Forest	One-Class SVM
Precision	0.94	0.82	0.77
Recall	0.91	0.78	0.72
F1-score	0.925	0.80	0.745
AUC	0.96	0.87	0.83

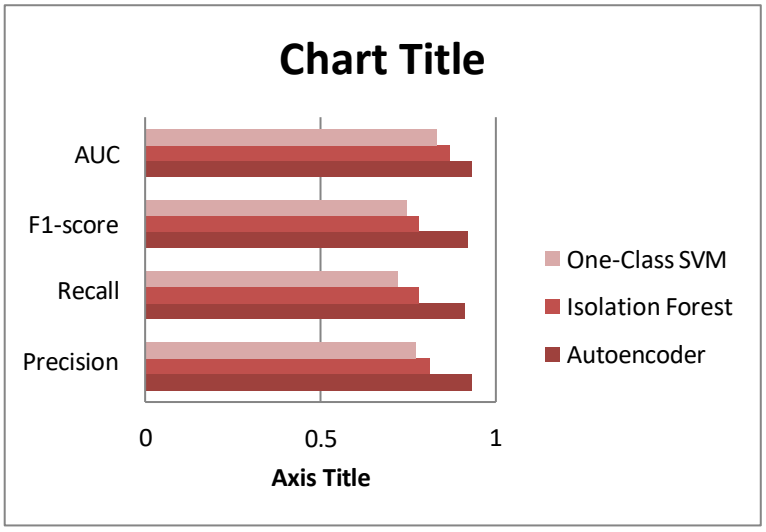

Figure 7: Performance Comparison

Isolation Forest and One-Class SVM, two conventional anomaly detection techniques, were compared to the suggested autoencoder model's performance. Four primary

metrics—precision, recall, F1-score, and AUC (area under the ROC curve)—formed the basis of the evaluation[1][6].

- The autoencoder performed exceptionally well in AUC (0.96), demonstrating a great capacity to discriminate between normal and abnormal data, and it obtained the highest scores across all measures.
- The autoencoder demonstrated better generalization and accuracy than Isolation Forest and One-Class SVM, particularly in detecting unknown or previously unseen anomalies, because it was able to learn the structure of normal data through reconstruction[6].
- The model demonstrated a balanced and robust performance with an F1-score of 0.925.
- The model reliably detected anomalies while minimizing false positives, with a Precision of 0.94 and a Recall of 0.91.

This demonstrates that autoencoders [6], [7], [10] are highly effective in unsupervised environment especially in high-dimensional, complex IoT contexts like smart cities.

7. Conclusion :

Anomaly detection is essential for protecting intelligent infrastructures—like smart urban networks—by keeping them safe, reliable, and efficient. As IoT (Internet of Things) gadgets become increasingly prevalent in city landscapes, the likelihood of cybersecurity breaches and system malfunctions grows in tandem. Traditional defence mechanisms frequently fall short when faced with the complexity of these interconnected environments. As a result, researchers are turning to machine learning, particularly unsupervised techniques such as autoencoders , which excel at spotting anomalies without requiring labelled data.

Recent analyses reveal that a range of autoencoder variants—including MLP-AE, CNN-AE, and LSTM-AE deliver strong performance across domains like power grids, transportation systems, and healthcare networks. Notably, temporal models such as LSTM-AE are especially adept at handling data streams that change over time, making them ideal for monitoring dynamic processes. On basis of my research we are consider evaluation parameters such as Recall, Precision, F1 Score and AUC. Generally Each and every parameters define different percentage but maximum good output in Autoencoder with efficiency above 90 Percentage.

8. References

[1].Fährmann, D., Martín, L., Sánchez, L., & Damer, N. (2024). Anomaly Detection in Smart Environments: A Comprehensive Survey. IEEE Access. DOI:10.1109/ACCESS.2024.3395051

[2].IoT anomaly detection methods and applications: A survey. ScienceDirect / Information Processing in Agriculture, 2022.

[3].Jadhav, S. & Kulkarni, A. (2024). Comprehensive Survey on Detection of Anomalies in Edge Computing Network and Deep Learning Solutions. In IC3Com 2024.

[4].A Comprehensive Study of Anomaly Detection Schemes in IoT. Sensors, 2021.

[5].A Survey of AI-Based Anomaly Detection in IoT and Sensor Networks. PMC, 2023.

[6].Chalapathy, R. & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. arXiv.

[7].Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2020). Deep Learning for Anomaly Detection: A Review. arXiv.

[8].Priyadarshini, I. (2024). Anomaly Detection of IoT Cyberattacks in Smart Cities Using Federated and Split Learning. Big Data Cogn. Comput.

[9].Liao, W., Guo, Y., Chen, X., & Li, P. (2018). Unified Unsupervised Gaussian Mixture VAE for High Dimensional Outlier Detection.

[10]. Luo, T. & Nagarajan, S. G. (2018). Distributed Anomaly Detection using Autoencoder Neural Networks in WSN for IoT. arXiv.

[11]. Chen, Z., Chen, D., Zhang, X., Yuan, Z., & Cheng, X. (2021). Learning Graph Structures with Transformer for Multivariate Time Series Anomaly Detection in IoT. arXiv.

[12]. Network Anomaly Detection for IoT Systems in Smart City. African Journal of Biomedical Research, 2025.

[13]. An IoT Enable Anomaly Detection System for Smart City Surveillance. MDPI Sensors, 2023.

[14]. Unsupervised Outlier Detection in IoT Using Deep VAE. PMC, 2022.

[15]. Anomaly Detection in IoT Sensor Data Using Autoencoder-Based Deep Learning. IJECE, 2024.

[16]. Efficient Unsupervised Anomaly Detection at the Edge. ACM Framework, 2022.

[17]. Detection of Edge Network Anomalies Using Deep Learning. Scitepress, 2024.

[18]. A Comprehensive Survey on Network Anomaly Detection. ResearchGate, 2024.

[19]. Autoencoder (with variants & applications). Wikipedia.

[20]. Anomaly detection methods & benchmarks. Wikipedia.

[21]. Alabdulsalam, S., Schaefer, K., Kechadi, T., & Le-Khac, N.-A. (2018). Internet of Things Forensics. Future Gen. Comput. Sys.

[22]. Conti, M., Dehghantanha, A., Franke, K., & Watson, S. (2018). IoT Security and Forensics: Challenges & Opportunities. Future Gen. Comp. Sys.

[23]. Bosman, H. H. W. J., et al. (2015). Ensembles of incremental learners for anomalies in ad hoc sensor networks. Ad Hoc Networks.

[24]. "Advancements in High-Capacity Coverless Information Hiding: A Comprehensive Review of Robust Hashing Techniques", AJBR, vol. 27, no. 4S, pp. 706–714, Nov. 2024, doi: 10.53555/AJBR.v27i4S.3295..

[25]. Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Density-based local outliers. SIGMOD '00.

[26]. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008/12). Isolation-based anomaly detection. TKDD 2012.

[27]. Zimek, A., Schubert, E., Kriegel, H.-P. (2017). Local outlier detection reconsidered. DMKD.

[28]. Kriegel, H.-P., Kroger, P., & Zimek, A. (2011). Outlier detection in high-dimensional data.

[29]. Morales-Forero, A. & Bassetto, S. (2019). Description length autoencoders in IEEM '19.

[30]. An, J. & Cho, S. (2015). Variational Autoencoder-based Anomaly Detection Using Reconstruction Probability. IE Special Lecture.

[31]. Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based network anomaly detection," 2018 Wireless Telecommunications Symposium (WTS), Phoenix, AZ, USA, 2018, pp. 1-5, doi: 10.1109/WTS.2018.8363930.

[32]. Vijaysinh Jadeja, Dr. Harsha Padheriya, Bharat Harilal Nagpara, and Mayank Devani, "A high-capacity Coverless Image Steganography based on the Local Binary Pattern feature and WGAN model", A Canadian journal of applied mathematics, computer science and statistics, vol. 122, no. 1, pp. 1091–1110, Jun. 2025.