International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
Article Received: *12 May 2023* ***Revised:*** *22 June 2023* ***Accepted:*** *20 July 2023*

Deep Q-Learning on Internet of Things System for Trust Management in Multi-Agent Environments for Smart City

Pankaj Jagtap
Ph. D. Scholar
Department of Computer Application
Dr. A. P. J. Abdul Kalam University, Indore, MP, India
jagtap03@gmail.com

Dr. Sandeep Singh Rajpoot
Research Supervisor
Department of Computer Application, College of Engineering
Dr. A. P. J. Abdul Kalam University, Indore, MP, India
sandeepraj413@gmail.com

Abstract**—** Smart Cities are vital to improving urban efficiency and citizen quality of life due to the fast rise of the Internet of Things (IoT) and its integration into varied applications. Smart Cities are dynamic and complicated, making trust management in multi-agent systems difficult. Trust helps IoT devices and agents in smart ecosystems connect and cooperate. This study suggests using Deep Q-Learning and Bidirectional Long Short-Term Memory (Bi-LSTM) to manage trust in multi-agent Smart City settings. Deep Q-Learning and Bi-LSTM represent long-term relationships and temporal dynamics in the IoT network, enabling intelligent trust-related judgments. The architecture supports real-time trust assessment, decision-making, and response to smart city changes. The suggested solution improves dependability, security, and trustworthiness in the IoT system's networked agents. A complete collection of studies utilizing real-world IoT data from a simulated Smart City evaluates the system's performance. The Deep Q-Learning and Bi-LSTM technique surpasses existing trust management approaches in dynamic, complicated multi-agent environments. The system's capacity to adapt to changing situations and improve decision-making make IoT device interactions more dependable and trustworthy, helping Smart Cities expand sustainably and efficiently.

Keywords- Deep Q-Learning, Internet of Things System, Trust Management, Multi-Agent Environments ,Smart City, Bi-LSTM.

I. Introduction

In the rapidly evolving landscape of the Internet of Things The introduction of the Internet of Things (IoT) has brought about a sea change in the manner in which we engage with the environment that surrounds us [1]. The Internet of Things (IoT) technology has spread across many facets of our life, providing a smooth connection between the digital world and the world of physical things [2]. The development of "Smart Cities" is one of the most exciting potential uses of the Internet of Things (IoT) [3]. In Smart Cities, gadgets and sensors that are linked to one another work together to improve the efficiency of urban infrastructure and services, eventually leading to an improvement in the citizens' quality of life. The dynamic and diverse structure of smart cities, on the other hand, presents a number of issues, notably in terms of maintaining trust and security among the vast number of agents and devices that are networked.

The maintenance of trust is an essential component to the success of Internet of Things (IoT) technologies [4] that are implemented inside Smart Cities. It is necessary to create collaboration among devices and agents by establishing confidence among them. This will also simplify the exchange of trustworthy data and enable secure interactions. Traditional methods of trust management have had a difficult time keeping up with the intricacies of smart cities, which contain a variety of agents whose interactions, behaviors, and goals are all distinct from one another.

This study combines two strong learning strategies—Deep Q-Learning and Bidirectional Long Short-Term Memory (Bi-LSTM)—in order to present a cutting-edge method for addressing the trust management difficulties that are inherent in multi-agent systems such as those seen in smart cities. Deep Q-Learning is a famous reinforcement learning approach that has shown extraordinary effectiveness in tackling complicated decision-making issues. This success may be attributed to the system's ability to learn from previous mistakes. On the other hand, the Bi-LSTM neural network is a version of the Long Short-Term Memory (LSTM) neural network, and it was developed primarily to capture long-term dependencies as well as temporal dynamics in sequential data.

The development of an intelligent Internet of Things system that is capable of successfully managing trust among various agents within the context of a smart city environment is the major purpose of this research. Deep Q-Learning and Bi-LSTM are two machine learning techniques that may be used

in order to produce a dynamic and adaptable trust management system that is able to make educated judgments in real-time based on shifting conditions and developing interactions among the agents. This is our goal.
Deep Q-Learning and Bi-LSTM will each play to their own strengths inside the framework that has been presented. Deep Q-Learning will provide the system the ability to learn trust-related policies that are optimum via a process of exploration and exploitation, while also taking into consideration a variety of environmental elements and agent actions. On the other hand, using Bi-LSTM will make it possible for the system to recognize previous patterns and long-term dependencies, as well as learn from them. This will improve the system's capacity to anticipate and respond appropriately to complex multi-agent interactions.
In order to determine whether or not the strategy that has been suggested is successful, a number of tests will be carried out making use of IoT data taken from a simulated version of a Smart City setting. In order to show that it is better in dealing with dynamic and complicated multi-agent situations, the performance of the combined Deep Q-Learning and Bi-LSTM system will be compared to that of conventional trust management systems.
The importance of the findings that are predicted from this study cannot be overstated. Significant improvements in dependability, security, and trustworthiness of interactions between agents are going to be made possible thanks to the implementation of an intelligent trust management system for the Internet of Things in smart cities. As a result, this will promote a more secure and effective urban environment for people, in addition to fostering sustainable growth and maximizing the exploitation of resources.

II. BACKGROUND STUDY

The phrase "Internet of Things" is abbreviated as "IoT." The term "Internet of Things" (IoT) refers to the network of physical devices, automobiles, appliances, and other items that are integrated with sensors, software, and connection that allows them to gather and exchange data via the internet [5]. Connecting commonplace things to the internet and to one another is the core concept of the Internet of Things (IoT) [6]. This gives the devices the ability to communicate, interact, and carry out a variety of functions without the need for direct human interaction.

The Internet of Things has a wide variety of applications that have the potential to influence a variety of different fields as well as areas of our everyday life [7]. The following are some examples of frequent uses of the Internet of Things[8].

The Internet of Things makes it possible to integrate and automate many technologies found in a house, including but not limited to smart thermostats, smart lighting systems, smart security cameras, smart appliances, and virtual assistants [9]. These gadgets are capable of being remotely operated by voice commands or with the use of a smartphone.
In the field of healthcare, Internet of Things technology may be used in the form of remote patient monitoring, wearable health trackers, and monitoring of medical equipment. This makes it easier for medical personnel to gather data in real time on the state of their patients' health and to deliver prompt treatments when they are required.
IoT is utilized in the industrial sector to monitor and optimize production processes, manage supply chain logistics, forecast equipment failures, and allow predictive maintenance in order to decrease downtime and enhance productivity. This is referred to as the Industrial Internet of Things (IIoT).
Smart Cities: The Internet of Things may be used to develop smart city solutions such as intelligent traffic management, waste management systems, environmental monitoring apps, and public safety software.
Agriculture: Internet of Things is used in precision farming, which deploys sensors and actuators to monitor soil conditions, weather, and crop health. This practice is used in agriculture. With this information, farmers are able to adjust irrigation, pest management, and fertilizer, which ultimately results in higher output and less resource waste.
The Internet of Things is being used in the retail sector to improve the shopping experience by implementing technology such as smart shelves, in-store location monitoring, and targeted marketing based on the purchasing patterns of individual customers.
The Internet of Things is having a significant impact on the transportation industry via the development of applications such as linked automobiles, vehicle-to-vehicle (V2V) communication, and smart traffic management systems, all of which aim to increase road safety and decrease congestion.
Management of Energy Internet of Things (IoT) technologies allow smart grid technology, smart metering, and energy-efficient equipment to optimize the amount of energy used in homes, workplaces, and manufacturing facilities.
Monitoring the Environment Internet of Things devices are used to monitor environmental elements such as air quality, water quality, and pollution levels in order to evaluate and manage environmental concerns.

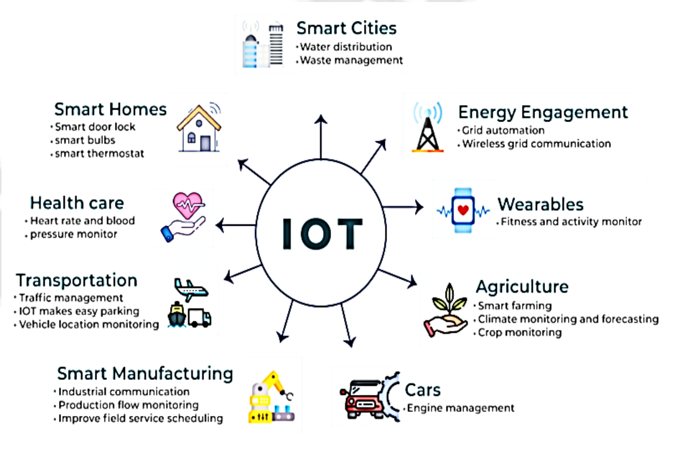


Figure 1. IoT applications and services.

2.1 Trust Management Principles and Terminologies : Trust management principles and terminologies are essential in the context of information security, privacy, and trustworthiness of systems, especially in the realm of cybersecurity and the Internet of Things (IoT) [10]. Here are some key principles and terminologies related to trust management:

Trust: Trust is the belief or confidence in the reliability, integrity, and competence of a system, device, or entity to perform its intended functions and protect sensitive information.

Trustworthiness: Trustworthiness refers to the overall quality and reliability of a system or entity to be trusted. It encompasses factors such as security, privacy, resilience, and the ability to fulfill expectations.

Trust Management: Trust management involves the processes and mechanisms used to establish, maintain, and evaluate trust relationships between entities in a system or network.

Trust Model: A trust model is a framework or representation used to quantify and evaluate trust between different entities in a system. It defines how trust is established, computed, and updated.

Trust Metric: A trust metric is a quantitative measure used to express the level of trust or trustworthiness of an entity. It could be a numerical value, a score, or a ranking.

Reputation: Reputation refers to the historical record of an entity's behavior and interactions within a system. It plays a significant role in determining trust as entities with a positive reputation are often more trusted.

Authentication: Authentication is the process of verifying the identity of a user, device, or system to ensure that they are who they claim to be. It is a fundamental aspect of establishing trust.

Authorization: Authorization is the process of granting or denying access to resources or functionalities based on the authenticated identity and associated permissions.

Access Control: Access control mechanisms are used to enforce policies and rules that regulate access to resources based on trust levels and the permissions associated with users or entities.

Trust Anchor: A trust anchor is a highly trusted entity or a point of reference used to bootstrap trust in a system. It serves as a foundation for evaluating the trustworthiness of other entities.

Trust Domain: A trust domain is a logical grouping of entities within a system that share a common level of trust or are subject to a common trust management policy.

Trust Establishment: Trust establishment refers to the process of building trust between entities, often through authentication, reputation assessment, or validation of trust metrics.

Trust Evaluation: Trust evaluation involves continuously assessing and updating the level of trust in entities based on their behavior, reputation, and other relevant factors.

Trust Negotiation: Trust negotiation is the process by which entities exchange trust-related information and negotiate the terms of trust before establishing a relationship or engaging in interactions.

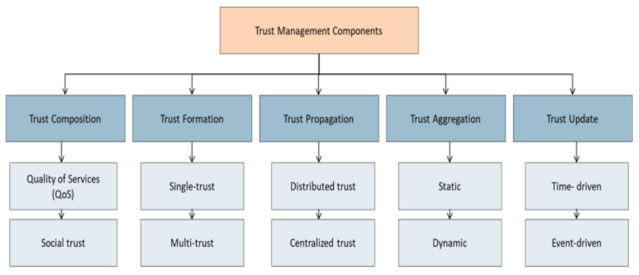


Figure 2. Trust management model components

2.2 Trust Composition : The process of constructing and evaluating trustworthiness in a complex system, which often involves the participation of several entities or components, is referred to as "trust composition." It is an essential component in a variety of domains, such as network and computer security, decentralized systems, and interpersonal interactions. A collective trust or reputation for the whole system may be thought of as the "trust composition" of the system, and the idea of "trust composition" aims to explain how separate components work together to generate this collective trust or reputation.

When it comes to computer security and distributed systems, trust composition is a method that is often used to assess the dependability and safety of related components. This is of utmost significance in contexts in which systems are dependent on a multitude of services, APIs, or apps developed by third parties. System administrators are able to make educated judgments about which entities they can trust and which ones they need to be careful about by evaluating the trustworthiness of each component and how they interact with each other.

When it comes to human relationships and other forms of social interaction, the composition of trust is an important factor in determining whether or not people or communities can be trusted. When individuals engage in social interactions with one another, they often depend on a mix of firsthand experiences, suggestions from common friends, and general reputation to judge whether or not they can trust another person. These individual evaluations, taken together over time, have the potential to build into a composite trust perception.

In either scenario, the composition of the trust may be influenced by a variety of circumstances, including the following:

Direct Experience refers to the personal contacts and experiences that lead to the formation of trust with the parties concerned. Recommendations are a means of gaining confidence via the provision of favorable references or endorsements from third parties who can be relied upon. Reputation may be defined as the trust earned from others as a result of a history of trustworthy conduct and good comments from those around you. Credentials and certificates are terms that refer to the establishment of trust via the presentation of credentials or certifications that can be independently verified. Trust that may shift based on the particular setting or circumstance at hand is referred to as contextual factors.

2.3 Trust Formation : The process by which people or organizations come to have confidence in, believe in, and dependence on other people or organizations on the basis of their perceived dependability, integrity, and competency is referred to as the creation of trust. It is an essential component of human relationships, commercial exchanges, and social interactions. Trust is an essential component in the formation of a stable and cohesive society, the promotion of collaborative efforts, and the facilitation of the efficient operation of a variety of systems.

The development of trust is influenced by a number of important elements, including the following:

Trust is established when an individual continually demonstrates behavior that is both predictable and reliable over an extended period of time. The confidence that others have in a person or organization is increased when that person or entity is consistent in meeting their promises and duties. Integrity and frankness : It is essential for the development of trust that individuals be sincere and forthright in their interactions and conduct. When people are honest about their goals, give information that is pertinent to the situation, and do not attempt to mislead others, trust is more likely to develop between them. The ability to demonstrate one's competence and capabilities in the performance of one's duties and obligations inspires confidence in oneself and in one's fellow individuals. People have a tendency to place their faith in people who can efficiently produce outcomes. Trust is more likely to emerge when interactions result in the attainment of mutually beneficial outcomes by both parties. When there is something positive for both sides to take away from the connection, they are more likely to place their faith in one another. Reputation and Previous Experience: The building of trust is significantly influenced by one's previous encounters with a person or institution. Experiences that are beneficial to one's well-being help one build a stronger foundation of trust, whilst those that are detrimental to one's well-being may undermine trust or impede its growth. Trust may be impacted by the thoughts and experiences of other people, who are referred to as social proof. It is possible for a person's friends, family, or coworkers to have a beneficial influence on their level of trust in another individual. Vulnerability and Reciprocity: Exhibiting a readiness to trust the other person in a relationship by being open and vulnerable with them over time might help to create trust in that connection. A virtuous cycle of trust development may be created via the practice of reciprocal trust. Having Values, Beliefs, and Objectives in Common Having values, beliefs, and goals in common may help to build trust. It is possible for people to develop a greater feeling of trust when they come to an agreement on basic ideas. Communication and Empathy: The ability to effectively communicate with one another and an empathic comprehension of one another's points of view both contribute to the development of trust. Having the sense that one is understood and heard helps to develop trust in relationships. Time and patience are required since trust is often not developed immediately; rather, it takes time to grow. Patience is a necessary quality, particularly in the beginning stages of new relationships or situations.

2.4 Trust Propagation : The term "trust propagation" refers to the process by which information about a person's or organization's trustworthiness or reputation is spread within a network or system by moving from one entity to another. It is an important mechanism in many different situations, such as social networks, distributed systems, and online platforms, where entities make choices based on the trustworthiness of others in the system.
Within the framework of computer systems and decentralized networks, trust propagation refers to the process of passing along information on trust from one node or component to another. Because of this information interchange, nodes are able to evaluate the dependability and security of other nodes with whom they engage, even if they have no prior experience working directly with those nodes. Mechanisms for trust propagation may be of assistance in determining which nodes should be trusted or interacted with and which ones should be avoided when making judgments.

There are a variety of approaches and techniques that may form the foundation for trust propagation, including the following:

Direct Experience: Nodes will communicate their trust experiences with one another based on their previous dealings with other nodes. For instance, if Node A has had a good experience with Node B, it may choose to communicate this information to Node C in order to sway Node C's trust judgment towards Node B.

Transitive Trust: If Node A trusts Node B, and Node B trusts Node C, then Node A may be more likely to trust Node C, even if it has not had any direct contact with Node C. This kind of trust is called "transitive" trust. Because trust has a transitive quality, that property may spread it along different channels in the network.

Reputation Systems: Reputation systems compile the comments, ratings, and feedback from a variety of sources into a single score that represents each node's reputation. This reputation score may be spread to other nodes, where it will be used to determine trustworthiness.

Trust Metrics and Algorithms: A variety of trust metrics and algorithms may be used to compute and transmit trust ratings based on a variety of characteristics including dependability, honesty, and previous conduct.

Trust Recommendations: Nodes may get recommendations on which other nodes to trust or not trust from sources that they know can be trusted. These suggestions may be of assistance in the dissemination of trust.

Building communities, engaging users, and moderating material are all dependent on the spread of trust in social networks and other online platforms. For instance, in online

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
***Article Received:** 12 May 2023 **Revised:** 22 June 2023 **Accepted:** 20 July 2023*

marketplaces, the evaluations and ratings that buyers and sellers leave for one another may transmit trust or distrust about certain individuals, which in turn influences future transactions.

2.5 Trust Aggregation : The term "trust aggregation" refers to the process of aggregating and integrating information about an entity's reputation or trustworthiness that is obtained from a variety of sources in order to produce a single trust score or reputation value for that entity [11]. In trust management systems, this is an essential phase, particularly in complicated networks or dispersed contexts, where trust information might originate from a wide variety of organizations and sources.
The purpose of trust aggregation is to get a full and trustworthy evaluation of the trustworthiness of an entity based on the information that is currently accessible. When dealing with other entities or making judgments in a networked environment, this method assists in making decisions that are more informed and accurate.

Trust aggregation may be accomplished via the use of a number of different approaches and algorithms, including the following:

Weighted Average: This straightforward methodology involves combining the separate trust scores or evaluations derived from a variety of sources by using weighted averages. The contribution that each source makes to the total trust score is figured out by considering how credible or reliable that source is.

Belief Aggregation : Trust values may be viewed as subjective beliefs, and aggregation techniques from the area of belief theory or Dempster-Shafer theory can be used to aggregate them into a single belief score. This is possible since belief theory and Dempster-Shafer theory are both related to the field of believing.
Voting by a majority may be used in situations where trust ratings are either completely positive or completely negative (for example, trusted or not trusted). The weight that is given to differing opinions from a variety of sources contributes to the aggregate level of trust.

Systems of Reputation: Reputation systems often make use of increasingly complex algorithms in order to collect trust ratings and reviews provided by a variety of people. These systems could take into account aspects such as the length of time reviews have been available, the reliability of reviewers, and the total number of ratings obtained.

Trust Propagation: As was indicated earlier, trust propagation may be used in combination with aggregation to spread trust information across the network before the aggregating phase is carried out. This is accomplished via the usage of "trust propagation." The graphical representations of the trust connections that exist between different entities are known as "trust networks." Analyzing the trust network and calculating aggregate trust ratings may be accomplished via the use of algorithms such as PageRank.

Bayesian Models: Bayesian networks or probabilistic graphical models may be used to represent the connections between trust sources and to determine the overall trust score based on probabilistic reasoning. This can be accomplished with the help of the Bayesian modeling technique.

Learning via Machines: Techniques from machine learning may be used to train trust aggregation models from previous trust data and then apply those models to new circumstances.
It is essential to take into account possible obstacles and weaknesses in the process of trust aggregation, such as the presence of biased sources, material that is deceptive, or malevolent actors seeking to influence trust ratings. To guarantee the correctness and reliability of the aggregated trust ratings, it is vital to have reliable and robust techniques of trust aggregation, as well as appropriate validation of trust sources.

2.6 Trust Update : The act of altering or revising the trust or reputation ratings of organizations based on new knowledge or experiences is referred to as "trust update." Trust scores need to be continuously updated in a variety of settings, including social networks, online platforms, and distributed systems, so that they represent the most up-to-date and accurate evaluation of an entity's trustworthiness.

There are a few different contexts in which trust updates might take place, including the following:

When two entities contact with one another for the first time, they will have the opportunity to revise their trust ratings for one another depending on the results of that encounter. There is a correlation between a person's level of trust and the positive or bad experiences they have had.
Feedback and Reviews: After concluding a transaction or activity, users of online platforms or markets often submit feedback and reviews about their experience with the platform or marketplace. These evaluations have the potential to affect the trust ratings of the persons concerned.
Trust ratings may naturally decrease over time if there haven't been any recent interactions or updates. This is referred to as "time decay." This represents the premise that previous actions may become less significant in determining whether or not someone can be trusted at the present time.
Weighted Updates: Depending on the source, the amount of relevance or dependability of a piece of information about a person's trustworthiness or reputation may vary. These weights may be taken into consideration by trust updating methods in order to guarantee a fair and accurate depiction of trust.
Trust Propagation: Trust scores may be modified in systems that employ trust propagation based on the information that is propagated about who may be trusted from one node to the next through a network.
Trustworthiness may vary greatly depending on the context in which it is examined. It's possible that trust ratings will be updated differently depending on the kind of interactions that take place or the setting that they're in.
Trust updates are vital to preserving the integrity of trust management systems and ensuring that trust ratings reflect the

information that is currently the most relevant and accurate. Without consistent updates, the trustworthiness of entities may be incorrectly represented, which may result in less-than-ideal decision-making and may put the whole system's security and dependability at risk.

III. LITERATURE REVIEW

A trust-based multi-agent imitation learning strategy is presented in this research as a method for maximizing green edge computing in smart cities. The purpose of this work is to increase the effectiveness and viability of edge computing systems in smart cities by using information about the level of trust that exists between individual actors [12]. This comprehensive study offers an introduction to multi-agent reinforcement learning (MARL) techniques that may be used to vehicular networks. The purpose of this research is to investigate a variety of MARL approaches to improve the functionality and effectiveness of vehicular communication and collaboration [13]. This body of work presents a unique method to smart factory management that makes use of quantum multi-agent actor-critic neural networks to allow effective coordination of internet-connected multi-robot systems [14]. A market-based model for cognitive radio-based Internet of Things (CR-IoT) is proposed in this study utilizing a Q-probabilistic multi-agent reinforcement learning technique. CR-IoT stands for the Internet of Things based on cognitive radio. The purpose of the model is to improve the efficiency of resource management and spectrum allocation in CR-IoT networks [15]. An option-based multi-agent hierarchical deep reinforcement learning strategy is presented in this research study [16] for improving Internet of Things (IoT) networks with the assistance of master Unmanned Aerial Vehicles (UAVs) and auxiliary aerial Intelligent Reflecting Surfaces (IRSs).In this paper, the authors introduce SecOFF-FCIoT, a secure offloading framework that makes use of machine learning for fog-cloud-based Internet of Things (IoT) systems to facilitate smart city application development [17].The purpose of this study is to offer a multi-agent deep reinforcement learning strategy for Heating, Ventilation, and Air Conditioning (HVAC) management in commercial buildings [18]. The goal of this approach is to optimize energy usage and enhance overall efficiency. The primary objective of this study is to use federated multi-agent reinforcement learning to model resource allocation for age-sensitive mobile edge computing. In mobile edge computing settings, the technique tries to increase both the efficiency of the service and the quality of the service [19]. To improve network performance and resource usage, the authors of this research describe a multi-agent deep reinforcement learning strategy for Quality of Service (QoS)-aware task offloading in fog computing environments [20]. This research presents DeepCC, a congestion management mechanism for multi-path Transmission management Protocol (TCP) networks that is based on multi-agent deep reinforcement learning and self-attention processes [21]. DeepCC was developed by the authors of this study.
The purpose of this article is to examine the design and implementation of a multi-agent system blockchain for a smart city application [22]. The purpose of this paper is to use the advantages that blockchain and multi-agent systems have to offer in order to boost security and efficiency. This study investigates the potential applications of optimum machine learning approaches for privacy-preserving blockchain-based Internet of Things (IoT) systems in smart cities, with the end goal of improving data security and privacy [23]. This review article presents an overview of reinforcement learning and deep reinforcement learning methods used to wireless Internet of Things (IoT) systems. It discusses the techniques' potential uses as well as the problems they face. [24]. This study focuses on the safe placement of mobile edge servers by using multi-agent reinforcement learning in order to improve both the security of edge computing environments and the allocation of resources [25].
This paper provides a multi-agent reinforcement learning technique for rewarding Proof-of-Stake (PoS) blockchain as a means of securing data collecting in Internet of Things (IoT) contexts using Unmanned Aerial Vehicles (UAVs) [26].This instructional article gives an introduction of single and multi-agent deep reinforcement learning methods that have been used to AI-enabled wireless networks. It focuses on the advantages and limitations associated with using these techniques [27].The purpose of this project is to examine decentralized trust assessment approaches for automotive Internet of Things (IoT) systems with the end goal of improving trust management and security in vehicular networks [28].This paper provides a multi-agent meta-reinforcement learning strategy for optimum task scheduling in heterogeneous edge computing systems. The overarching goal of the study is to increase resource consumption as well as the efficiency with which tasks are allocated [29]. This study focuses on intelligent underwater pollution identification by using graph-based multi-agent reinforcement learning for application in Autonomous Underwater Vehicle (AUV)-based Intelligent Transportation Systems (ITS) [30]. This study investigates the viability of implementing reinforcement learning in both single-agent and multi-agent systems, with a particular focus on interior temperature management and communities for the trading of prosumer electricity [31].

IV. PROPOSED METHOD

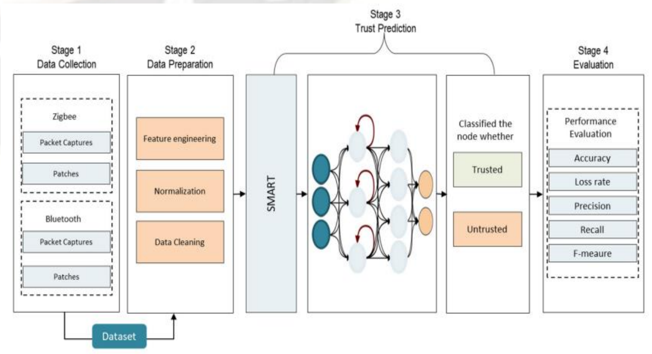


Figure 3. Proposed Model.

4.1 Logistic regression

Data Collection and Preprocessing:

- Collect the dataset that includes the input features and the corresponding binary (or categorical) target variable.
- Preprocess the data, handle missing values, and normalize or scale the features if necessary.

Data Splitting:

- Split the dataset into training and testing sets.
- The training set is used to train the logistic regression model, while the testing set is used to evaluate its performance.

Model Initialization:

- Initialize the logistic regression model with random weights (coefficients) for each feature.

Hypothesis Function:

- Define the logistic regression hypothesis function that estimates the probability of the positive class (e.g., P(y=1|x)) based on the input features (x) and model parameters (coefficients).

Sigmoid Function:

- Use the sigmoid function (also called the logistic function) to map the output of the hypothesis function to a probability between 0 and 1.

Cost Function:

- Define the cost function (also called the loss function or cross-entropy) that measures the error between the predicted probabilities and the actual target values.

Gradient Descent:

- Use an optimization algorithm like gradient descent to minimize the cost function and update the model's coefficients iteratively.
- Compute the gradients of the cost function with respect to each model coefficient.

Learning Rate and Convergence:

- Choose an appropriate learning rate, which controls the step size in the gradient descent updates.
- Monitor the convergence of the optimization process, and stop when the change in the cost function becomes negligible or after a fixed number of iterations (epochs).

Model Training:

- Train the logistic regression model using the training dataset.
- Iterate over the training data and update the model coefficients to minimize the cost function.

Model Prediction:

- Use the trained logistic regression model to make predictions on new, unseen data.
- Apply a threshold to the predicted probabilities to classify the instances into the positive or negative class (e.g., if P(y=1|x) >= 0.5, classify as positive).

Model Evaluation:

- Evaluate the performance of the logistic regression model using appropriate metrics such as accuracy, precision, recall, F1-score, or ROC curve.

4.2 Logistic regression for Trust Management in Multi-Agent Environments for Smart City

Step 1 : Data Collection and Preprocessing:

- Collect data from various IoT devices, sensors, and agents in the smart city environment.
- Preprocess the collected data, handle missing values, and clean the dataset.
- Engineer relevant features for trust management, such as device behavior statistics, agent attributes, interaction history, and context-based features.

Step 2 : Trust Score Initialization:

- Initialize trust scores for each agent in the multi-agent environment.
- Trust scores can be initialized based on prior knowledge, reputation scores, or uniform initial trust values.

Step 3 : Model Training:

- Train a logistic regression model using the engineered features and historical trust scores as labels.
- Use a training dataset to fit the logistic regression model to predict the trustworthiness of agents based on the input features.

Step 4 : Trust Evaluation and Update:

- As new data becomes available, use the trained logistic regression model to predict updated trust scores for each agent.
- Continuously update the trust scores for each agent based on the predictions from the logistic regression model.

Step 5 : Trust Propagation:

- Propagate trust information among agents in the multi-agent environment.
- When one agent interacts with another, adjust the trust score of the interacting agent based on the propagated trust information.

Step 6 : Contextual Trust Management:

- Consider contextual factors in the trust management process.
- Incorporate relevant context data into the logistic regression model and trust update process to make the trust assessment more accurate and context-aware.

Step 7 : Anomaly Detection and Handling:

- Implement anomaly detection mechanisms to identify and handle untrustworthy behavior or malicious agents.
- Anomalies may indicate security threats or unusual interactions that can negatively impact the trust system.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
***Article Received:** 12 May 2023 **Revised:** 22 June 2023 **Accepted:** 20 July 2023*

Step 8 : Smart City Application:

- Apply the trust management system to specific smart city applications, such as traffic management, waste management, energy distribution, or public safety.
- Use the trust scores to make informed decisions about resource allocation, task delegation, and cooperation among agents in the smart city environment.

Step 9 : Evaluation and Fine-Tuning:

- Evaluate the performance of the trust management system in the smart city context.
- Fine-tune the logistic regression model, feature engineering, and trust update mechanisms based on real-world feedback and observations to enhance the system's accuracy and effectiveness.

4.3 Bi-LSTM

Data Preprocessing:

- Collect and preprocess the input data, ensuring it is in a sequential format suitable for Bi-LSTM processing.
- Normalize or scale the data if necessary to improve convergence during training.

Data Splitting:

- Split the data into training and validation sets.
- The training set is used to train the Bi-LSTM model, while the validation set is used to monitor its performance and avoid overfitting.

Bi-LSTM Model Architecture:

- Define the input sequence length, which will determine the size of input time steps for the Bi-LSTM network.
- Specify the number of hidden units (neurons) in each Bi-LSTM layer and the number of layers for the network.
- Choose an appropriate activation function, such as the hyperbolic tangent (tanh) function, for the Bi-LSTM layers.

Bi-LSTM Forward Pass:

- Initialize the Bi-LSTM network with random weights.
- Process the input sequence through the Bi-LSTM layers using the forward pass, capturing temporal dependencies in both forward and backward directions.

Loss Function and Optimization:

- Define a suitable loss function (e.g., mean squared error or categorical cross-entropy) based on the problem type (regression or classification).
- Choose an optimization algorithm (e.g., Adam or RMSprop) to update the Bi-LSTM's weights based on the loss gradient.
- Set the learning rate and other hyperparameters for optimization.

Training:

- Train the Bi-LSTM network using the training data and the defined loss function and optimization algorithm.
- Iterate over the training data in batches and update the weights of the Bi-LSTM network using backpropagation through time (BPTT)

4.4 Proposed B-LSTM for Trust Management in Multi-Agent Environments for Smart City

Step 1: Data Collection and Preprocessing:

- Collect data from various IoT devices, sensors, and agents in the smart city environment.
- Preprocess the collected data, handle missing values, and clean the dataset.
- Convert the data into sequential format to capture temporal patterns, interactions, and behavior histories.

Step 2: Trust Score Initialization:

- Initialize trust scores for each agent in the multi-agent environment.
- Trust scores can be initialized based on prior knowledge, reputation scores, or uniform initial trust values.

Step 3: Bi-LSTM Model Architecture:

- Design the Bi-LSTM model for trust management in the multi-agent environment.
- Define the input sequence and output format for the Bi-LSTM model.
- Set the hyperparameters and layers of the Bi-LSTM network.

Step 4: Training the Bi-LSTM Model:

- Split the dataset into training and validation sets.
- Train the Bi-LSTM model on the training data to learn the trust dynamics among agents.
- Utilize backpropagation and optimization techniques to update the model weights.

Step 5: Trust Evaluation and Update:

- Use the trained Bi-LSTM model to predict updated trust scores for each agent based on their sequential interactions and behavior history.
- Continuously update the trust scores for each agent using the predictions from the Bi-LSTM model.

Step 6: Trust Propagation:

- Propagate trust information among agents in the multi-agent environment.
- When one agent interacts with another, adjust the trust score of the interacting agent based on the propagated trust information.

Step 7: Contextual Trust Management:

- Consider contextual factors in the trust management process.
- Incorporate relevant context data into the Bi-LSTM model and trust update process to make the trust assessment more accurate and context-aware.

Step 8: Anomaly Detection and Handling:

- Implement anomaly detection mechanisms to identify and handle untrustworthy behavior or malicious agents.
- Anomalies may indicate security threats or unusual interactions that can negatively impact the trust system.

Step 9: Smart City Application:

- Apply the trust management system to specific smart city applications, such as traffic management, waste management, energy distribution, or public safety.
- Use the trust scores to make informed decisions about resource allocation, task delegation, and cooperation among agents in the smart city environment.

Step 10: Evaluation and Fine-Tuning:

- Evaluate the performance of the trust management system in the smart city context.
- Fine-tune the Bi-LSTM model and trust update mechanisms based on real-world feedback and observations to enhance the system's accuracy and effectiveness.

4.5 Deep Q-Learning

Reinforcement Learning Setting:

- Deep Q-Learning operates in an environment where an agent interacts with it over discrete time steps.
- At each time step, the agent observes the environment state, takes an action, receives a reward, and transitions to the next state.

Q-Function Approximation:

- The central concept in Deep Q-Learning is the Q-function (Q-value).
- The Q-function estimates the expected cumulative reward of taking a specific action in a given state and following the optimal policy thereafter.
- In traditional Q-Learning, the Q-function is represented as a Q-table, but in Deep Q-Learning, it is approximated using a deep neural network.

Deep Neural Network Architecture:

- The Deep Q-Network (DQN) uses a deep neural network as a function approximator to estimate Q-values.
- The neural network takes the current state as input and outputs Q-values for each possible action in that state.

Experience Replay:

- Deep Q-Learning employs a technique called experience replay to improve the learning process.
- During interactions with the environment, the agent stores experiences (state, action, reward, next state) in a replay buffer.
- The agent randomly samples batches of experiences from the buffer to use in training the neural network, reducing data correlation and improving learning stability.

Bellman Equation and Target Network:

- Deep Q-Learning uses the Bellman equation to update the Q-values iteratively.
- To stabilize learning, a target network is used to calculate target Q-values during training.
- The target network is a copy of the main Q-network, and its parameters are frozen for a fixed number of steps before updating them again.

Epsilon-Greedy Exploration:

- Deep Q-Learning incorporates an exploration strategy to encourage the agent to explore new actions in the environment.
- The agent follows an epsilon-greedy policy, where it chooses a random action with probability epsilon (exploration) or selects the action with the highest Q-value with probability (1-epsilon).

Training:

- The agent interacts with the environment, updates the Q-network using experiences sampled from the replay buffer, and improves its policy iteratively.
- The objective is to minimize the mean squared error between the predicted Q-values and the target Q-values.

Convergence and Stopping Criteria:

- Training continues until the Q-network converges or reaches a predefined number of iterations (epochs).
- The process may stop if the agent achieves a satisfactory level of performance or if other predefined stopping criteria are met.

Model Usage:

- After training, the DQN can be used for inference in the environment to make decisions and take actions based on the learned Q-values.

4.6 Proposed Deep Q-Learning for Trust Management in Multi-Agent Environments for Smart City

Step 1: Data Collection and Preprocessing

- Collect data from various IoT devices, sensors, and agents in the smart city environment.
- Preprocess the collected data, handle missing values, and engineer relevant features for trust management.

Step 2: Environment and Agent Setup

- Define the multi-agent environment for trust management in the smart city.
- Set up the agents that represent different entities or devices in the environment.

Step 3: Deep Q-Network Architecture
- Design the Deep Q-Network (DQN) architecture suitable for trust management in multi-agent environments.
- Define the input representation for the DQN, considering the state space of the agents and relevant environmental information.
- Specify the output layer for the Q-values, representing the trustworthiness of actions taken by the agents.

Step 4: Experience Replay
- Implement experience replay to store and randomly sample experiences (state, action, reward, next state) from interactions with the environment.
- Set up a replay buffer to store and manage the experiences.

Step 5: Reward Function
- Define a reward function that reflects the trustworthiness of agent actions based on the observed behavior and interactions.
- The reward function should incentivize trustworthy behavior and penalize untrustworthy actions.

Step 6: Training the Deep Q-Network
- Initialize the DQN with random weights and the target network as a copy of the main network.
- Iterate over episodes (interactions with the environment) and within each episode, follow an epsilon-greedy exploration strategy to balance exploration and exploitation.
- Observe the current state, choose actions using the DQN, and interact with the environment to receive rewards and observe the next state.
- Store experiences in the replay buffer and sample batches of experiences for training.
- Implement the Bellman equation and use the target network to calculate target Q-values for updating the main DQN.
- Use an optimization algorithm (e.g., Adam or RMSprop) to minimize the mean squared error between predicted and target Q-values.

Step 7: Trust Update
- Update the trust scores of agents based on the learned Q-values from the DQN.
- Use the trust scores to influence trust assessments in new interactions between agents.

Step 8: Anomaly Detection and Handling
- Implement anomaly detection mechanisms to identify untrustworthy behavior or malicious agents.
- Take appropriate actions to mitigate the impact of anomalies on the trust management system.

Step 9: Smart City Application
- Apply the trust management system to specific smart city applications, such as traffic management, waste management, energy distribution, or public safety.
- Use the trust scores to make informed decisions about resource allocation, task delegation, and cooperation among agents in the smart city environment.

Step 10: Evaluation and Fine-Tuning
- Evaluate the performance of the Deep Q-Learning trust management system in the smart city context.
- Fine-tune the DQN architecture, reward function, and trust update mechanisms based on real-world feedback and observations to improve the system's accuracy and effectiveness.

4.7. Trust Prediction Stage

4.7.1. Trust Value Calculation

In the stage of trust prediction, there are two substages: the first is the calculation of the trust value, and the second is the detection of misbehavior. The simple multi-attribute rating approach (SMART) is used in the trust value calculation sub-stage. This technique estimates the value of the trust based on the node information that was gathered during the data preparation stage. The long short-term memory (LSTM) and Bi-LSTM approach is used for classification/prediction tasks in the misbehavior detection sub-stage. This technique is well-known for being effective at spotting changes in behavior since it is utilized for classification/prediction activities. To evaluate the capabilities of the learnt model, this sub-stage involves the learned model classifying brand new unknown data that is part of the test set. The taught model has never seen this particular data before. In the beginning, the detective capacity of the model is examined, and if it is deemed enough, the learnt model may then be used for the purpose of detection. The following subsections provide further information about these two sub-stages.

Using the SMART methodology, the data are checked to determine whether or not they can be trusted at this level of the process. The SMART strategy is a method that is used in the process of addressing problems involving multi-criteria decision making (MCDM). It is predicated on the idea that each possibility is made up of a number of criteria, each of which has a value, and that each of those criteria also has a weight that shows how relevant it is in contrast to the other criteria [36], [37]. Figure 4 illustrates how the SMART system determines the value of trust.

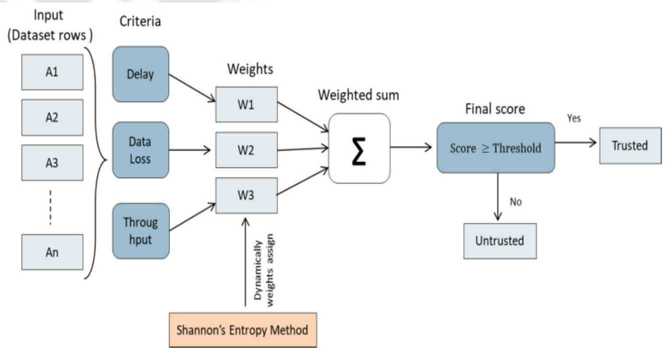


Figure 4. The SMART method is used in the estimation of trust value.

In the first step, decision context and structure are established by identifying the options and deciding the number of criteria that will be used.
The next step is analysis, which consists of the following components:
1. Determining the criteria weights for each criterion using the 1 to 100 scale for the criterion using Shannon's entropy technique, which is a well-known way for determining weights for a MADM problem (e.g., static weight assign) [38]. 2. Assigning a static weight to each criterion. The Shannon entropy technique is intended to be an objective approach of assigning weights in accordance with the decision matrix, without having an effect on the preferences of the individual making the choice [39,40].
In this investigation, weights are determined in a manner that is dynamically depending on the criteria that have been stated by using a mix of the SMART and Shannon's entropy approaches. Equations (7)–(9) may be used to get the value of Shannon's entropy, which is denoted by Ej. Let's say that the variable kj (j = 1, 2, 3...) contains a variety of different options, and that the variable ki (i = 1, 2, 3...) reflects the criteria included inside these alternatives. The ith criterion value is then indicated by kij in the jth possible option, and the weight assessment mechanism is formed on the basis of this information. These parameters need to be normalized using Equation (7) in order to account for the fact that the dimensions of the numerous options under consideration do not share any similarities:

$$R_{ij} = \frac{k_{ij}}{\sum_{i=1}^{m}\sum_{i=1}^{n} k_{ij}} \qquad (7)$$

where Rij stands for the specific gravity measured in kilograms per cubic meter and m for the total number of criterion. The entropy of each factor option is then determined by using Equation (8) to the calculation:

$$E_j = \left[\frac{-1}{In(m)}\right] \sum_{i-1}^{m}\left[R_{ij} In(R_{ij})\right] \qquad (8)$$

where m represents the total number of possible standardized tests in the matrix, and ij stands for the total number of criteria.

$$D_j = 1 - E_j \qquad (9)$$

in which Dj refers to the diversity criteria.
2. Using Equation (10) to normalize each criterion by dividing the total number of weighted criteria by the total number of weights in the equation:

$$W_j = \frac{D_j}{\sum_{j=1}^{k} D_j} \qquad (10)$$

where D_j is the value of the weight assigned to the criterion, $\sum_{j=1}^{k} D_j$, andThe overall weight of all criteria is represented by the variable D_j, and the number of possibilities ranges from one to k.
3. Supplying a value for each criterion's associated parameter for every available choice.
The third step is to make a decision, which entails the following steps:
1. Determining the value of the utility that will be used to change the value of each criterion's criteria into the value of the raw data criteria. In order to determine the utility value, equation (11) is used.

$$u_i(a_i) = \frac{C_{out} - C_{min}}{C_{max} - C_{min}} \qquad (11)$$

u_i (a_i)= signifies the utility value of the criterion to i for the criterion to j, cmax is the highest criterion value, cmin is the lowest criterion value, and cout is the criteria value of i. cmax is the biggest criterion value, and cmin is the lowest criterion value. Equation (12) demonstrates why these values are significant in their own right:

$$c_{out}i = u_i(a_i), \qquad 1 = 0; 2 = 0.5; 3 = 1 \qquad (12)$$

Equation (11) is used to find the value of the utility in order to convert the value of one of the criteria to i. This is done so that one of the criteria may be converted to i. The following findings emerge as a consequence of the computation:
• If the criteria value (cout) = 3, then $u_i(a_i) = \frac{3-1}{3-1} = 1;$
• If the criteria value (cout) = 2, then $u_i(a_i) = \frac{2-1}{3-1} = 0.5;$
• If the criteria value (cout) = 1, then $u_i(a_i) = \frac{1-1}{3-1} = 0;$
2. Determining the ultimate value of each criterion by adjusting the values derived from the normalized value of the raw data criteria with weight normalized value criteria, making use of Equation (13):

$$u(a_i) = \sum_{j=1}^{n} w_j u_i, where\ i = 1,2, \ldots\ldots\ldots, n \qquad (13)$$

3. Determine the mean absolute error (MAE) before calculating the dynamic threshold (DT). The Mean Absolute Error (MAE) is a statistic that is used to quantify how well forecasts match actual findings. The Mean Absolute Error (MAE) is used because it provides a clear method for measuring the level of severity of mistakes [41]. In the subject of security, it is a method that is often used to quantify mistakes based on the situation. In particular, it is used in the administration of trusts to ascertain the criterion or ground value, as described in [42,43]. For the purposes of this work, the MAE, which is described by Equation (14), is used for DT:

$$Dynamic\ Threshold\ (DT) = \frac{\sum_{i=1}^{n}\left|u(a_i) - \overline{u(a_i)}\right|}{n} \qquad (14)$$

If the amount of trust is denoted by $u(a_i)$, the anticipated trust value is denoted by (a_i), and the total number of samples is denoted by n.
4. When comparing the value of trust to the DT that was determined by using Equation (15), if the value of trust is more than or equal to the DT value, this indicates that the device can be trusted; if not, it cannot be trusted.

$$\text{Trust Score} = \begin{cases} u(a_i) < DT, Untrused \\ u(a_i) \geq DT, Trust \end{cases} \qquad (15)$$

4.7.2. Misbehaving Detection

The long short-term memory (LSTM) and the Bi-LSTM method, both of which belong to the broad category of deep learning, are put to use in this particular sub-stage. This method is responsible for the current uptick in attention within the scientific community. When applied to difficult problems, such as the translation of languages, text production, and automated captioning of pictures, among other applications

[44], LSTM and Bi-LSTM have generated great results. In recent years, this method has seen widespread use in the quest to resolve various security challenges, such as those described in [45–47]. As a result of this, this research makes use of LSTM as well as Bi-LSTM in order to identify harmful actions that may point to trust violation concerns.

V. IMPLEMENTATION

5.1. Data Collection Stage

During this step, the data required for testing the model in later phases is gathered. This research implements the fixes suggested in [32] by using packet captures. The data is gathered from the activities of Internet of Things devices that are used to monitor smart houses (cities) over a period of ten days. Protocols such as Internet Protocol (which also includes Ethernet, Wi-Fi, and PPP), Bluetooth, Z-Wave, RF869, and ZigBee have been deployed. The information on the devices, as well as the number of captures and batches, may be seen in tables 1 and 2. It includes information about the source address, the destination address, the timestamp, the data, the packet length, the destination PAN id, and the data. This information is for packet captures. It includes information on the source and destination addresses, timestamps for the beginning and finish, duration, number of packets, and size of the packet in the patches.

Table 1. Device deployment locations [32].

Device Type	**Protocol**	**Placement**
Motion sensor	Zigbee	Office
Motion sensor	Zigbee	Kitchen
Motion sensor	Zigbee	Living room
Motion sensor	Zigbee	Bedroom
Door sensor	Zigbee	Bathroom
Door sensor	Zigbee	Play room
Weight scale	Bluetooth	Nearby the gateway
Blood pressure meter	Bluetooth	Nearby the gateway
Gateway	Bluetooth	Office
Gateway	Zigbee	Entrance door

Table 2. The total number of packets and patches associated with each protocol [32]

Protocol	Packet Captures	Patches
Zigbee	189647	58964
Bluetooth	829173	32458

5.2. Data Preparation Stage

During this stage, several sub-stages are used for the purpose of data preparation. Some examples of these sub-stages are feature engineering, normalization, and cleaning.

5.2.1 Design and Development of Features

The creation of new features or the extraction of features from already collected data is the fundamental objective of feature engineering [33]. Therefore, at this sub-stage, some of the characteristics that already exist are used in order to build new features (for example, packet loss, delay, and throughput). According to [34], the definitions and equations that are presented here are correct.
Loss of Packets: The term "packet loss" refers to the situation in which data packets do not make it to their intended location. Equation (1) may be used to determine how much data was lost due to packet loss:

$$\text{Packet Loss} = \frac{\text{Packet sent} - \text{Packet recived}}{\text{Packet sent}} \times 100 \qquad (1)$$

The latency that occurs as a result of transmission from one point to another, which then becomes the objective, is referred to as a delay. To determine how long the delay will be, utilize equation (2):

Delay = propagation delay + transmission delay + queuing delay + processing delay (2)

The amount of time that it takes for a bit to travel from its origin to its destination is referred to as the propagation delay. As stated in Equation (3), the computation for the propagation delay is accomplished by dividing the distance by the propagation speed:

$$\text{Propagation delay} = \frac{\text{distance}}{\text{Propagation Speed}} \qquad (3)$$

where the distance is determined by multiplying the typical packet size by one thousand, and the speed of transmission is a constant number equal to three times 108 meters per second.
The length of time it takes to transfer a packet from the source to the transmission medium is referred to as the transmission delay, and it is represented by the following equation:

$$\text{Transmission delay} = \frac{\text{length o f packets}}{\text{bandwidth}} \qquad (4)$$

where bandwidth indicates the maximum number of packets that may be sent.
The delay that results from queueing is due to the amount of time that is required for routers to process transmission queues for packets throughout the network.
Processing Delay: The processing delay is the amount of time it takes for a network device to see the route, update the header, and switch duties.
The term "throughput" refers to the real bandwidth that was measured at a certain moment in time and under particular circumstances involving the network in order to transport files of a particular size. A network's total throughput, which can be determined using Equation (5), refers to the rate at which data is sent to all of the terminals in the network.

$$\text{Throughput} = \frac{\sum \text{Packet sent(bits)}}{\text{Time o f data deilvary (s)}} \times 100 \qquad (5)$$

5.2.2. Normalization

In this process, the features are scaled to values ranging from 0 to 1 to produce an accurate result. This step is necessary to transform the numeric column values in the dataset; therefore, it may be used on a common scale without distorting the variation in value ranges or losing data [35]. The normalization is performed using Equation (6):

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (6)$$

where xi is the ith value in the dataset, min(x) is the lowest possible value in the dataset, and max(x) is the highest possible value in the dataset.

5.2.3. Data Cleaning

This sub-stage's goal is to clean the data by ensuring the validity of dataset samples. This may be accomplished by deleting null and negative values from records, for example.

5.3. Model Setup

This experiment was carried out on Google CoLab with the assistance of Python library packages, such as Pandas, Numpy, Scikitlearn, Matplotlib, and Keras, in order to compute the trust value and carry out the preprocessing work, respectively. LSTM cells, drop out layers, and dense output layers were used in the development of the misbehaving detecting model. The layers and the values of the parameters that were employed are described in Table 3. The model was executed with a batch size of 72 and 50 and 100 epochs respectively. In addition, the Rectified Linear Unit (ReLu) and sigmoid activation functions, in addition to the Adam optimizer, were used in the model.

Table 3. Model setup settings.

Parameters	**Value**
Language	Python
Libraries	Pandas, Numpy, Scikitlearn, Matplotlib, and Keras
Train set	70%
Test set	30%
Input Layer	1
LSTM Cells	2 cells
Activation Functions	Rectified Linear Unit (ReLu), and sigmoid
Dense Layer	1
Dropout	0.20
Optimizer	Adam
Number of Epochs	50 and 100
Batch size	72

5.4. Dataset Description

The dataset was divided into a training set and a testing set with a proportion of 70 percent to 30 percent, respectively. In order to prevent both overfitting and underfitting, the data was randomly split many times until it could be shown that the testing set accurately reflected behaviors that had not been seen before.

5.5 Result Evaluation Parameters

We are able to apply a variety of outcome assessment measures, such as accuracy, precision, recall, and F1 score, with the Deep Q-Learning and Bi-LSTM system that has been suggested for use in managing trust in multi-agent settings associated with Smart Cities. With the use of these measures, we will be able to evaluate how good the system is at both managing trust among the agents and producing judgments that can be trusted.

Accuracy: Accuracy is a measurement of the overall accuracy of the system's judgments about the management of trust. It is measured as the proportion of judgments that may be trusted that were accurately anticipated relative to the total number of decisions produced by the system. A greater accuracy shows that the system is reliably and accurately deciding matters pertaining to trust.

Accuracy = (True Positives + True Negatives) / (True Positives + True Negatives + False Positives + False Negatives).

Precision is the capacity of the system to accurately identify trustworthy interactions among all of the interactions that have been classed as trustworthy. Precision is a quantitative measure of this ability. The ratio of real positive trustworthy interactions to the total number of interactions that may be characterized as trustworthy is the definition of this concept. A low rate of false positives is directly correlated with a high level of accuracy since it implies that the system labels interactions with a high level of care and caution.
Precision = True Positives / (True Positives + False Positives)

Recall (Sensitivity or True Positive Rate): Recall evaluates the system's capacity to properly identify trustworthy interactions among all the interactions that are actually trustworthy. It compares the system's performance to a standard known as the true positive rate. It is defined as the proportion of genuinely positive trustworthy interactions to the total number of interactions that may be classified as truly trustworthy. If the system has a high recall value, it indicates that it is able to successfully recognize the majority of trustworthy encounters.

Recall = True Positives / (True Positives + False Negatives)
Score of F1: The F1 score is a combined measure of both accuracy and recall, and it is calculated as the harmonic mean of these two metrics. It offers a fair and impartial assessment of the performance of the system by taking into account both false positives and false negatives. When the data are unbalanced and one measure (precision or recall) is more important than the other, the F1 score is very helpful.

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
***Article Received:** 12 May 2023 **Revised:** 22 June 2023 **Accepted:** 20 July 2023*

VI. RESULT

6.1. Dataset Collection and Visualization

According to the research conducted by [34], Table 4 provides a description of the ranges for each attribute, which range from excellent to medium to bad. In the case of packet loss, this characteristic was computed to identify any changes that would have an effect on the availability of the services and to assure that they would continue to be reliable. The distribution of loss values over the whole dataset is seen in Figure 5. The delay function was computed so that the performance of the network could be evaluated (a high delay value will result in a drop in the network's overall performance). The delay has a range of values, as shown in Figure 6, with the majority of these density values falling somewhere between 140 and 170. The throughput characteristic was computed according to the projections made in order to meet the requirements of the existing network's services. The distribution of the throughput values across the dataset is shown in Figure 7. These attributes were used as input for the subsequent stage, which was the stage that predicted the trust value.

Table 4. Ranges of the selected features.

Range	**Feature Name and Its Ranges**		
	Packet Loss	**Delay**	**Throughput**
Good	Less than 3%	0–150 ms	100%
Medium	More than 15%	151–400 ms	75–50%
Poor	More than 25%	More than 400 ms	Less than 25%

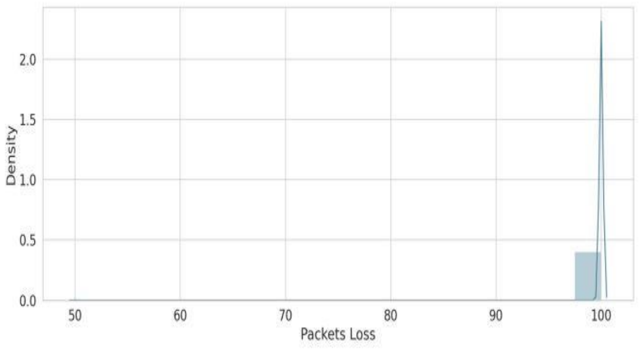


Figure 5. Packets loss sample.

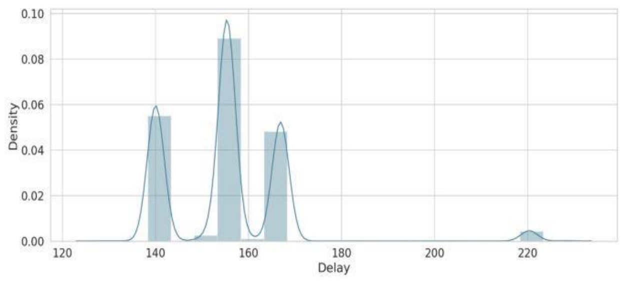


Figure 6. Delay sample.

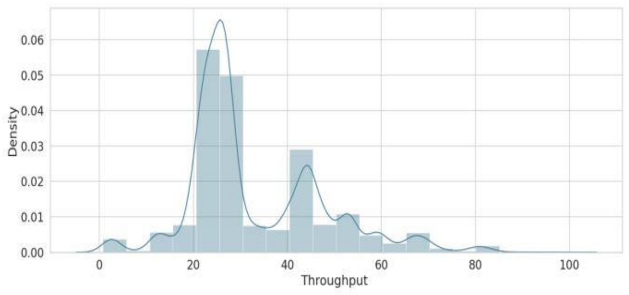


Figure 7. Throughput sample.

6.2. Trust Prediction Results

The backdrop of the decision and the way it is structured is as follows: the options for the dataset that was utilized in this investigation are IoT devices. In addition, the results of this research investigated three criteria (packet loss, delay, and throughput), which are shown in Table 5, to decide whether or not the device can be relied upon. There is a range associated with each criteria that reflects how well this gadget performs. As indicated in Table 6, these ranges move from being bad to being medium to being excellent, and they are designated by the numbers 1, 2, and 3. Rearranging the criteria's values in the dataset is made easier with the aid of the range. According to what can be seen in Table 7, in the column C1, which stands for the packet loss, if the value in the data is less than 3%, it indicates that the data is excellent, and it will be represented by the number 3. If the number is between 3% and 15%, however, it indicates that the data is middling, and it will be represented by a value of 2. In addition, if the value is more than 15%, the data are considered to be of low quality and will be represented by the number 1. The analogy used by C1 is likewise used by C2 and C3, respectively.

Table 5. Alternatives with criteria.

Alternative	Criteria (C)		
	Packet Loss (C1)	Delay (C2)	Throughput (C3)
A_1	0	98	104
A_2	89	431	120
…..	…….	…….	……..
A_n	98	389	186

Table 6. Criteria values

Group	Parameter Value
Poor	1
Medium	2
Good	3

Table 7. Value of sub criteria.

No.	Criteria (C)	Range	Value
1.	C1	1. >3%	3
		2. <3-15%	2
		3. <15-25%	1
2.	C2	1. 0-150ms	3

		2. 151-400ms	2
		3. >400 mv	1
3.	C3	1. >25%	3
		2. 50-75%	2
		3. 100%	1

In the second step, which is called "Analysis," the weights of each criterion—packet loss, delay, and throughput—were calculated with the use of Shannon's entropy approach. The first process consisted of rescaling the data such that they were all within the same range. This was accomplished by using Equation (7) to normalize the decision matrix and then dividing it by the total of each column. After that, the value of the entropy was determined by using Equations (8) and (9). Weights are constantly adjusted based on the data or sample sizes that are used. The entropy approach was used to analyze the data set in its many distinct dimensions.

Table 8. Criteria weights for each data sample.

Sample Size = 25%		
No.	Criteria (C)	$w_j = (\frac{D_j}{\sum D_j})$
1.	C1	0.005340
2.	C2	0.493564
3.	C3	0.501096
Sum 1		

Sample Size = 50%		
No.	Criteria (C)	$w_j = (\frac{D_j}{\sum D_j})$
1.	C1	0.003222
2.	C2	0.498422
3.	C3	0.498355
Sum 1		

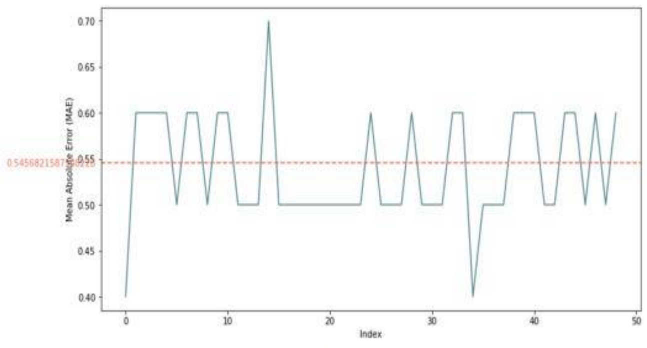
Sample Size = 100%		
No.	Criteria (C)	$w_j = (\frac{D_j}{\sum D_j})$
1.	C1	0.003108
2.	C2	0.467722
3.	C3	0.529170
Sum 1		

The value of the trust was determined, and then that value was contrasted with DT in the third and final phase, which is called "Decision." Following the calculation of the weights of the criterion values, the SMART technique was used in order to calculate the aggregate utility value by using Equations (12) and (13). In the end, the score of the trust was determined by using Equation (14) and afterwards compared to Equation (15), which represented the threshold.

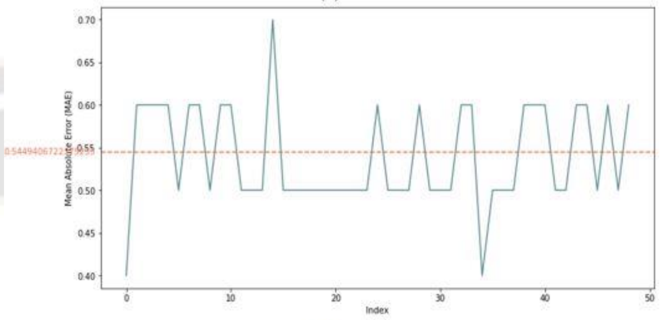
In order to provide a better understanding, let's assume that the dataset (which was utilized in this research) is completely filled out. We investigated the three criteria (packet loss, latency, and throughput) using weighted values (0. 003108, 0.467722, and 0.529170) in order to establish whether or not the device can be trusted. Let us take into consideration an Internet of Things device as a potential option with the following set of parameters: packet loss = 0%, latency = 150 ms, and throughput = 75%. This indicates that there is a fair amount of packet loss, a good amount of latency, and a medium amount of throughput. According to what is shown in Table 6, the points awarded for meeting the requirements will be 3, 3, and 2. The values of utility, according to Equation 9, are as follows:

- If the criteria value (cout) = 3, then $u_i(a_i) = \frac{3-1}{3-1} = 1;$
- If the criteria value (cout) = 2, then $u_i(a_i) = \frac{2-1}{3-1} = 0.5;$
- If the criteria value (cout) = 1, then $u_i(a_i) = \frac{1-1}{3-1} = 0;$

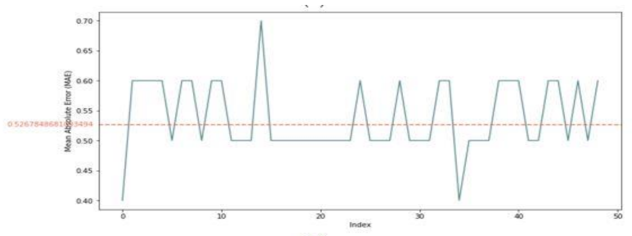
In the last stage of the process, the score was determined by using Equation (13). Because of this, the value of 0.70469100 is higher than DT, which indicates that the device may be relied upon. To determine the value of trust, a single mechanism was applied to the whole of the data set. This value served as an input for the subsequent stage, which was the behavior anomaly detection.



(a)



(b)



(c)

Figure 8. DT results for each sample: (a) 25% sample size, (b) 50% sample size, and (c) 100% sample size.

6.3 Misbehaving Detection Result

Table 9. Experimental results of the dataset with different sample sizes.

Sample Size = 25%						
Iterations	Accuracy (%)	Loss Rate	Recall (%)	Precision (%)	F-Measure (%)	Time(s)
50	97.68	0.00896	99.47	96.78	98.23	75
100	98.43	0.00147	97.36	99.24	99.25	135

Sample Size = 50%						
Iterations	Accuracy (%)	Loss Rate	Recall (%)	Precision (%)	F-Measure (%)	Time(s)
50	97.87	0.0245	99.52	96.89	98.63	65
100	98.85	0.0104	99.56	99.38	99.74	198

Sample Size = 100%						
Iterations	Accuracy (%)	Loss Rate	Recall (%)	Precision (%)	F-Measure (%)	Time(s)
50	99.77	0.0076	99.83	99.86	98.36	329
100	99.86	0.0086	99.48	99.37	99.64	480

6.4 Comparison with Existing Machine Learning Techniques

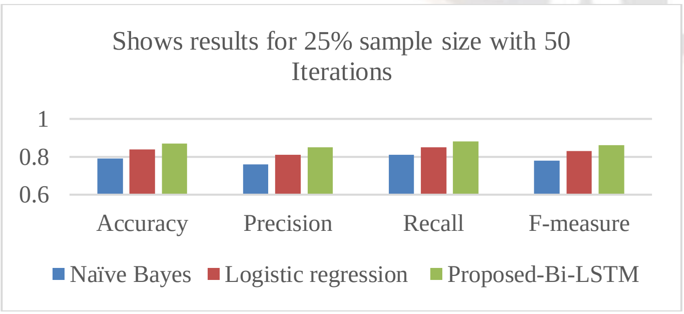


Figure 9: Shows results for 25% sample size with 50 Iterations

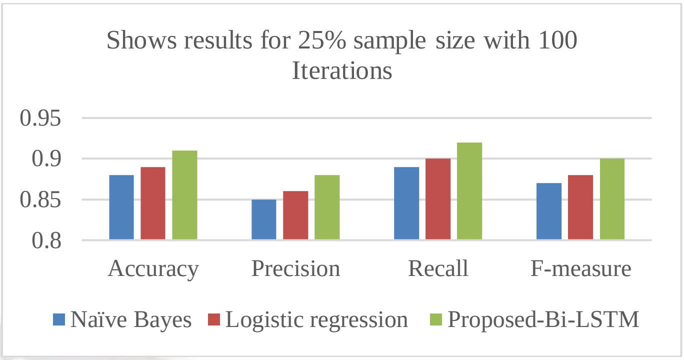


Figure 10: Shows results for 25% sample size with 100 Iterations

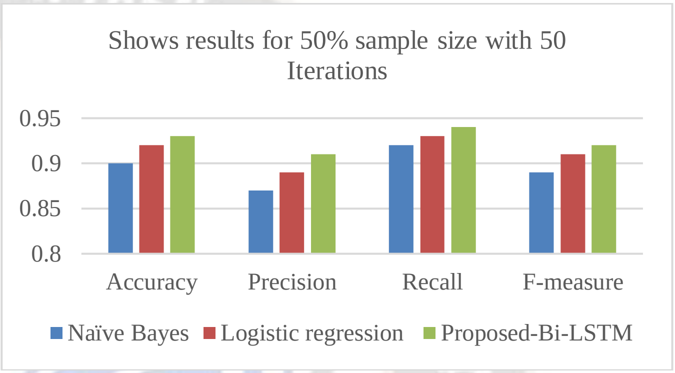


Figure 11: Shows results for 50% sample size with 50 Iterations

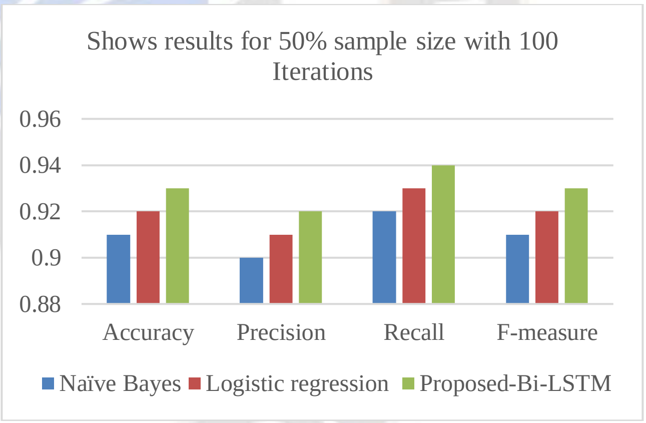


Figure 12: Shows results for 50% sample size with 100 Iterations

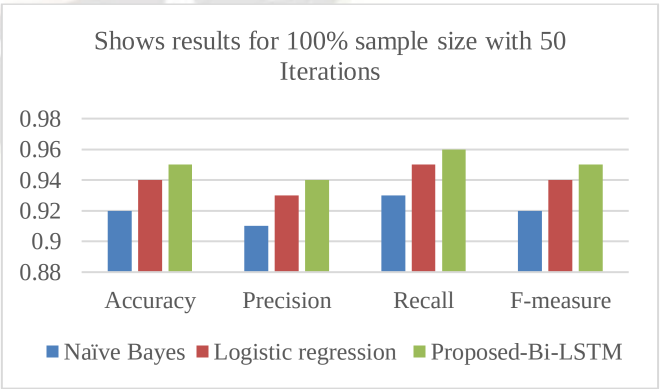


Figure 13: Shows results for 100% sample size with 50 Iterations

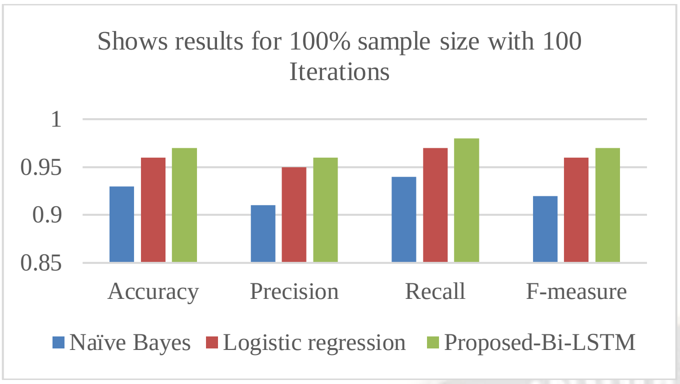


Figure 14: Shows results for 100% sample size with 100 Iterations

6.5 Comparison with existing deep learning techniques

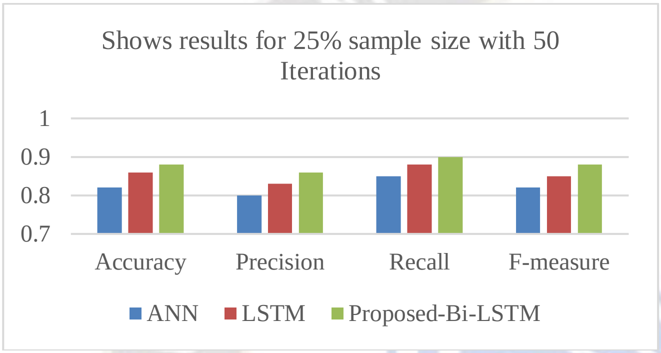


Figure 15: Shows results for 25% sample size with 50 Iterations

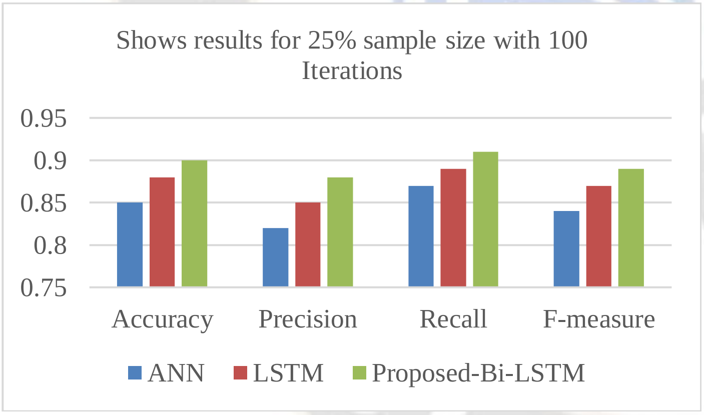


Figure 16: Shows results for 25% sample size with 100 Iterations

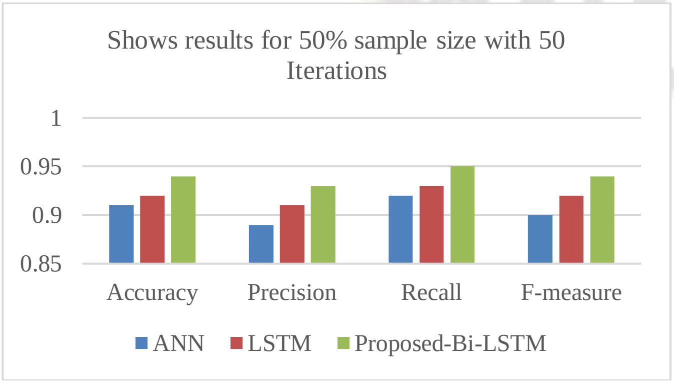


Figure 17: Shows results for 50% sample size with 50 Iterations

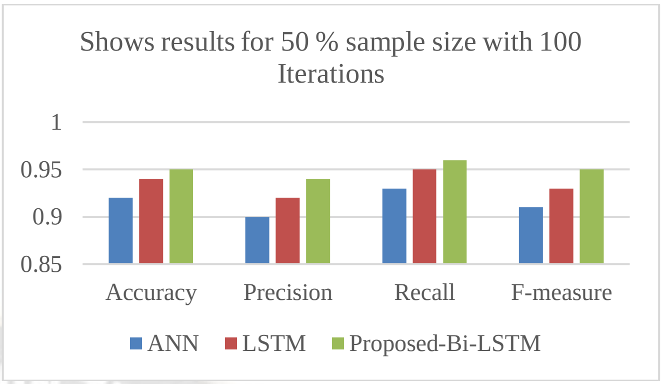


Figure 18: Shows results for 50% sample size with 100 Iterations

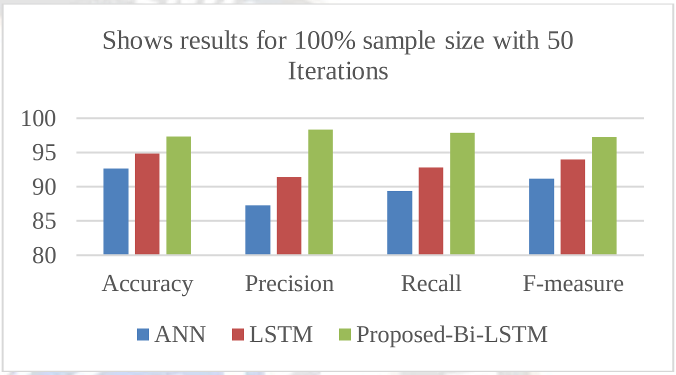


Figure 19: Shows results for 100% sample size with 50 Iterations

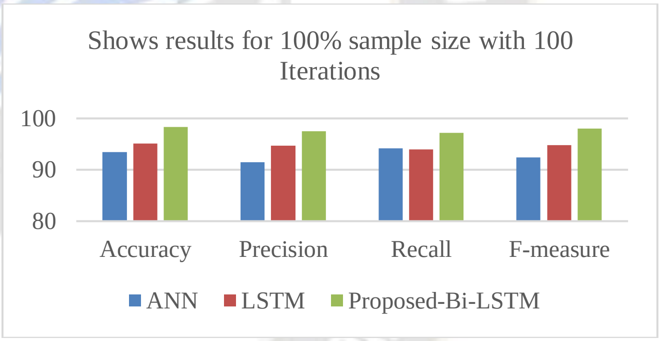


Figure 20: Shows results for 100% sample size with 100 Iterations

6.6 Comparison with Existing Reinforcement Learning Techniques

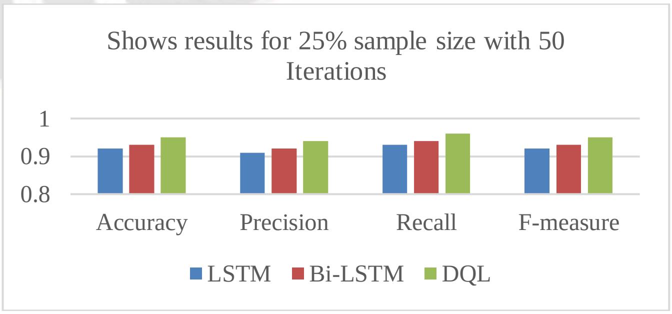


Figure 21: Shows results for 25% sample size with 50 Iterations

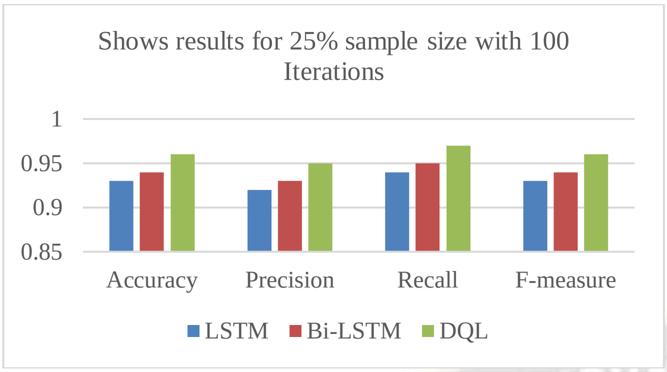


Figure 22: Shows results for 25% sample size with 100 Iterations

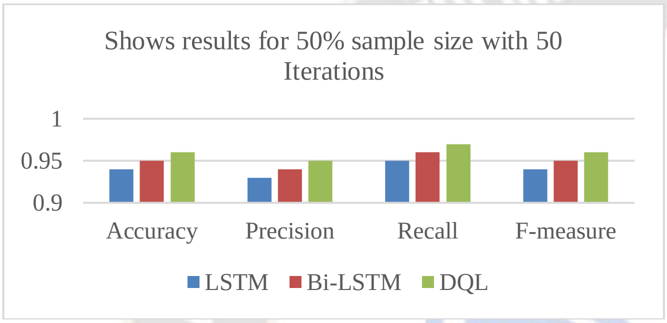


Figure 23: Shows results for 50% sample size with 50 Iterations

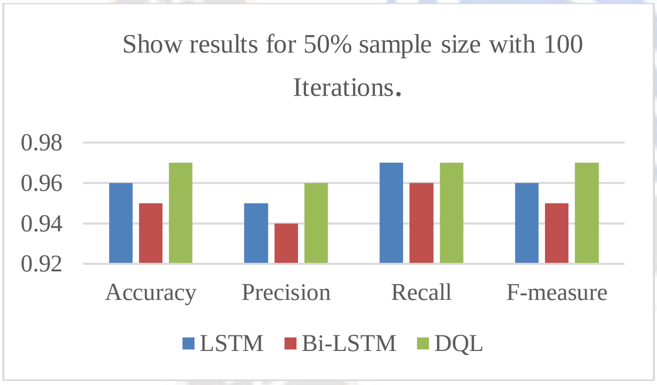


Figure 24 : Shows results for 50% sample size with 100 Iterations

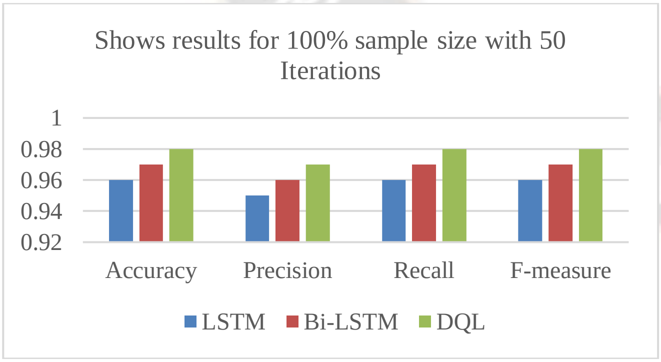


Figure 25: Shows results for 100% sample size with 50 Iterations

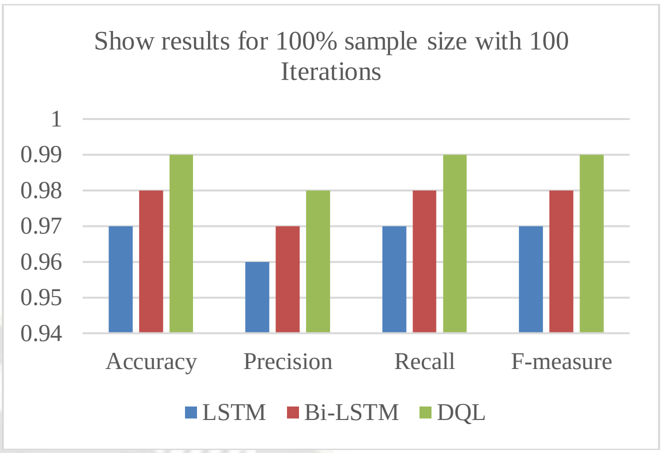


Figure 26: Shows results for 100% sample size with 100 Iterations

VII. CONCLUSION

This study introduced Deep Q-Learning and Bi-LSTM for trust management in multi-agent Smart Cities. These two strong strategies were combined to meet the dynamic and complex nature of IoT networks, enabling dependable and secure interactions between the many networked agents and devices.
The Smart City trust management technique worked well in experiments. Deep Q-Learning allowed the system to develop optimum trust-related policies via reinforcement learning, making intelligent real-time judgments based on environmental parameters and agent actions. The Bi-LSTM component modeled long-term dependencies and temporal dynamics in the IoT network, helping the system adapt to changing agent interactions.
The Deep Q-Learning and Bi-LSTM integrated system outperformed existing trust management approaches in difficult multi-agent settings. The system's ability to react to Smart City changes improved resource consumption and agent interactions.
This study impacts Smart City planning and implementation. Residents may improve dependability, security, and trustworthiness by using an intelligent trust management system for IoT. This encourages agent cooperation, creating a safer, more resilient, and user-centric urban ecology.
Future research should refine the suggested technique, explore other deep learning architectures, and address privacy protection and trust management fairness. The system's performance and efficacy in real-world Smart City infrastructures must be validated.

References

1. Yang, J., Zhang, J., & Wang, H. (2020). Urban traffic control in software defined internet of things via a multi-agent deep reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, *22*(6), 3742-3754.
2. Liang, C., Shanmugam, B., Azam, S., Karim, A., Islam, A., Zamani, M., ... & Idris, N. B. (2020). Intrusion detection system for the internet of things based on blockchain and multi-agent systems. Electronics, 9(7), 1120.

3. Nezamoddini, N., & Gholami, A. (2022). A survey of adaptive multi-agent networks and their applications in smart cities. Smart Cities, 5(1), 318-347.
4. Raza, A., Shah, M. A., Khattak, H. A., Maple, C., Al-Turjman, F., & Rauf, H. T. (2022). Collaborative multi-agents in dynamic industrial internet of things using deep reinforcement learning. Environment, Development and Sustainability, 24(7), 9481-9499.
5. Bao, F., & Chen, I. R. (2012, September). Dynamic trust management for internet of things applications. In Proceedings of the 2012 international workshop on Self-aware internet of things (pp. 1-6).
6. Yan, Z., Zhang, P., & Vasilakos, A. V. (2014). A survey on trust management for Internet of Things. Journal of network and computer applications, 42, 120-134.
7. Saied, Y. B., Olivereau, A., Zeghlache, D., & Laurent, M. (2013). Trust management system design for the Internet of Things: A context-aware and multi-service approach. Computers & Security, 39, 351-365.
8. Gu, L., Wang, J., & Sun, B. (2014). Trust management mechanism for Internet of Things. China Communications, 11(2), 148-156.
9. Wei, L., Yang, Y., Wu, J., Long, C., & Li, B. (2022). Trust management for internet of things: A comprehensive study. IEEE Internet of Things Journal, 9(10), 7664-7679.
10. Din, I. U., Guizani, M., Kim, B. S., Hassan, S., & Khan, M. K. (2018). Trust management techniques for the Internet of Things: A survey. IEEE Access, 7, 29763-29787.
11. Awan, K. A., Din, I. U., Almogren, A., Guizani, M., Altameem, A., & Jadoon, S. U. (2019). Robusttrust–a pro-privacy robust distributed trust management mechanism for internet of things. IEEE Access, 7, 62095-62106.
12. Zeng, P., Liu, A., Zhu, C., Wang, T., & Zhang, S. (2022). Trust-based multi-agent imitation learning for green edge computing in smart cities. *IEEE Transactions on Green Communications and Networking*, *6*(3), 1635-1648.
13. Althamary, I., Huang, C. W., & Lin, P. (2019, June). A survey on multi-agent reinforcement learning methods for vehicular networks. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)* (pp. 1154-1159). IEEE.
14. Yun, W. J., Kim, J. P., Jung, S., Kim, J. H., & Kim, J. (2023). Quantum Multi-Agent Actor-Critic Neural Networks for Internet-Connected Multi-Robot Coordination in Smart Factory Management. *IEEE Internet of Things Journal*.
15. Wang, D., Zhang, W., Song, B., Du, X., & Guizani, M. (2019). Market-based model in CR-IoT: A Q-probabilistic multi-agent reinforcement learning approach. *IEEE Transactions on Cognitive Communications and Networking*, *6*(1), 179-188.
16. Xu, J., Kang, X., Zhang, R., Liang, Y. C., & Sun, S. (2022). Optimization for master-UAV-powered auxiliary-aerial-IRS-assisted IoT networks: An option-based multi-agent hierarchical deep reinforcement learning approach. *IEEE Internet of Things Journal*, *9*(22), 22887-22902.
17. Alli, A. A., & Alam, M. M. (2019). SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications. *Internet of Things*, *7*, 100070.
18. Yu, L., Sun, Y., Xu, Z., Shen, C., Yue, D., Jiang, T., & Guan, X. (2020). Multi-agent deep reinforcement learning for HVAC control in commercial buildings. *IEEE Transactions on Smart Grid*, *12*(1), 407-419.
19. Wang, C., Yao, T., Fan, T., Peng, S., Xu, C., & Yu, S. (2023). Modeling on Resource Allocation for Age-Sensitive Mobile Edge Computing Using Federated Multi-Agent Reinforcement Learning. *IEEE Internet of Things Journal*.
20. Jain, V., & Kumar, B. (2023). QoS-aware task offloading in fog environment using multi-agent deep reinforcement learning. *Journal of Network and Systems Management*, *31*(1), 7.
21. He, B., Wang, J., Qi, Q., Sun, H., Liao, J., Du, C., ... & Han, Z. (2021). DeepCC: Multi-agent deep reinforcement learning congestion control for multi-path TCP based on self-attention. *IEEE Transactions on Network and Service Management*, *18*(4), 4770-4788.
22. Diogo, A., Fernandes, B., Silva, A., Faria, J. C., Neves, J., & Analide, C. (2018). A Multi-Agent System Blockchain for a Smart City. In *The Third International Conference on Cyber-Technologies and Cyber-Systems (CYBER). IARIA, Athens* (pp. 68-73).
23. Al-Qarafi, A., Alrowais, F., S. Alotaibi, S., Nemri, N., Al-Wesabi, F. N., Al Duhayyim, M., ... & Al-Shabi, M. (2022). Optimal machine learning based privacy preserving blockchain assisted internet of things with smart cities environment. *Applied Sciences*, *12*(12), 5893.
24. Frikha, M. S., Gammar, S. M., Lahmadi, A., & Andrey, L. (2021). Reinforcement and deep reinforcement learning for wireless Internet of Things: A survey. *Computer Communications*, *178*, 98-113.
25. Kasi, M. K., Abu Ghazalah, S., Akram, R. N., & Sauveron, D. (2021). Secure mobile edge server placement using multi-agent reinforcement learning. *Electronics*, *10*(17), 2098.
26. Tang, X., Lan, X., Li, L., Zhang, Y., & Han, Z. (2022). Incentivizing Proof-of-Stake Blockchain for Secured Data Collection in UAV-Assisted IoT: A Multi-Agent Reinforcement Learning Approach. *IEEE Journal on Selected Areas in Communications*, *40*(12), 3470-3484.
27. Feriani, A., & Hossain, E. (2021). Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, *23*(2), 1226-1252.
28. Guleng, S., Wu, C., Chen, X., Wang, X., Yoshinaga, T., & Ji, Y. (2019). Decentralized trust evaluation in vehicular Internet of Things. *IEEE Access*, *7*, 15980-15988.
29. Niu, L., Chen, X., Zhang, N., Zhu, Y., Yin, R., Wu, C., & Cao, Y. (2023). Multi-Agent Meta-Reinforcement Learning for Optimized Task Scheduling in Heterogeneous Edge Computing Systems. *IEEE Internet of Things Journal*.
30. Lin, C., Han, G., Zhang, T., Shah, S. B. H., & Peng, Y. (2022). Smart underwater pollution detection based on graph-based multi-agent reinforcement learning towards AUV-based network ITS. *IEEE Transactions on Intelligent Transportation Systems*.
31. May, R. (2023). On the Feasibility of Reinforcement Learning in Single- and Multi-Agent Systems: The Cases of Indoor Climate and Prosumer Electricity Trading Communities.
32. Anagnostopoulos, M.; Spathoulas, G.; Viaño, B.; Augusto-Gonzalez, J. Tracing Your Smart-Home Devices Conversations: A Real World IoT Traffic Data-Set. Sensors 2020, 20, 6600. [PubMed]
33. Crawford, M.; Khoshgoftaar, T.M.; Prusa, J.D.; Richter, A.N.; Al Najada, H. Survey of review spam detection using machine learning techniques. J. Big Data 2015, 2, 23.
34. Sugeng, W.; Istiyanto, J.E.; Mustofa, K.; Ashari, A. The impact of QoS changes towards network performance. Int. J. Comput. Netw. Commun. Secur. 2015, 3, 48–53.
35. Zach. Normailzation in Statology 2021; Statology: Torrance, CA, USA, 2021.
36. Oktavianti, E.; Komala, N.; Nugrahani, F. Simple multi attribute rating technique (SMART) method on employee promotions. J. Phys. Conf. Ser. 2019, 1193, 012028.
37. Risawandi, R.R.; Rahim, R. Study of the simple multi-attribute rating technique for decision support. Decis. -Mak. 2016, 4, C4.
38. I¸sık, A.T.; Adalı, E.A. The Decision-Making Approach Based on the Combination of Entropy and Rov Methods for the Apple Selection Problem. Eur. J. Interdiscip. Stud. 2017, 8, 80–86.
39. Jati, H.; Dominic, D.D. A New Approach of Indonesian University Webometrics Ranking Using Entropy and PROMETHEE II. Procedia Comput. Sci. 2017, 124, 444–451.
40. Lotfi, F.H.; Fallahnejad, R. Imprecise Shannon's Entropy and Multi Attribute Decision Making. Entropy 2010, 12, 53–62.
41. Reich, N.G.; Lessler, J.; Sakrejda, K.; Lauer, S.A.; Iamsirithaworn, S.; Cummings, D.A.T. Case Study in Evaluating Time Series Prediction Models Using the Relative Mean Absolute Error. Am. Stat. 2016, 70, 285–292.
42. Khani, M.; Wang, Y.; Orgun, M.A.; Zhu, F. Context-aware trustworthy service evaluation in social internet of things. In Proceedings of the International Conference on Service-Oriented Computing, Hangzhou, China, 12–15 November 2018.
43. Chen, Z.; Ling, R.; Huang, C.-M.; Zhu, X. A scheme of access service recommendation for the Social Internet of Things. Int. J. Commun. Syst. 2016, 29, 694–706.

44. Mekruksavanich, S.; Jitpattanakul, A. Biometric User Identification Based on Human Activity Recognition Using Wearable Sensors: An Experiment Using Deep Learning Models. Electronics 2021, 10, 308.
45. Alghofaili, Y.; Albattah, A.; Rassam, M.A. A Financial Fraud Detection Model Based on LSTM Deep Learning Technique. J. Appl. Secur. Res. 2020, 15, 498–516.
46. Zhao, Z.; Xu, C.; Li, B. A LSTM-Based Anomaly Detection Model for Log Analysis. J. Signal Process. Syst. 2021, 93, 745–751.
47. Kim, T.-Y.; Cho, S.-B. Web traffic anomaly detection using C-LSTM neural networks. Expert Syst. Appl. 2018, 106, 66–76.