Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 5, pp. 3778–3795, DOI 10.12694/scpe.v25i5.3004

RECURRENT NEURAL NETWORK BASED INCREMENTAL MODEL FOR INTRUSION DETECTION SYSTEM IN IOT

HIMANSHU SHARMA, PRABHAT KUMAR[*] AND KAVITA SHARMA[†]

Abstract. The security of Internet of Things (IoT) networks has become a integral problem in view of the exponential growth of IoT devices. Intrusion detection and prevention is an approach ,used to identify, analyze, and block cyber threats to protect IoT from unauthorized access or attacks. This paper introduces an adaptive and incremental intrusion detection and prevention system based on RNNs, to the ever changing field of IoT security. IoT networks require advanced intrusion detection systems that can identify emerging threats because of their various and dynamic data sources. The complexity of IoT network data makes it difficult for traditional intrusion detection techniques to detect potential threats. Using the capabilities of RNNs, a model for creating and deploying an intrusion detection and prevention system (IDPS) is proposed in this paper. RNNs work particularly well for sequential data processing, which makes them an appropriate choice for IoT network traffic monitoring. NSL-KDD dataset is taken, pre-processed, features are extracted, and RNN-based model is built as a part of the proposed work. The experimental findings illustrate how effective the suggested approach is at identifying and blocking intrusions in Internet of Things networks. This paper not only demonstrates the effectiveness of RNNs in enhancing IoT network security but also opens avenues for further exploration in this burgeoning field. It presents a scalable, adaptive intrusion detection and prevention solution, responding to the evolving landscape of IoT security. As IoT networks continue to expand, the research enriches the discourse on developing resilient security strategies to combat emerging threats in scalable computing environments.

Key words: IoT, IDS, Machine Learning, Deep Learning, RNN

1. Introduction.

Internet of Things is a network that allows everyday electronic devices to exchange data and coordinate their actions. The level of interconnection holds out the possibility of greater ease and efficiency in our day-to-day activities. Nevertheless, just as there are two sides to every coin, there are considerable worries associated with the Internet of Things (IoT), notably in regard to its security. The significance of IoT networks cannot be overstated; that has the potential to transform industries, improve the quality of life, and drive economic growth.

IoT networks are driving innovation in industrial automation, making manufacturing processes more efficient and reducing downtime [1]. Industries are using connected machines to streamline their operations and reduce downtime. In transportation, they are paving the way for autonomous vehicles, which have the potential to revolutionize mobility and reduce accidents. In the realm of energy, IoT enables the smart grid, optimizing energy distribution and promoting energy efficiency. In smart cities, IoT facilitates the creation of urban environments where infrastructure, transportation, and utilities are interconnected [2]. Cities are becoming smarter by embedding sensors that can help manage traffic in real-time, turn off streetlights when no one's around, or even alert about potential infrastructure issues. This promises sustainability, reduced traffic congestion, and an improved quality of life for urban residents. In agriculture, precision farming driven by IoT allows farmers to optimize crop yields, conserve resources, and promote sustainable practices, addressing the global challenge of food security. In the realm of healthcare, IoT devices enable remote patient monitoring, personalized treatment plans, and timely interventions. Patients can receive better care, and healthcare providers can operate more efficiently.

IoT devices often have limited computational resources and may need robust security mechanisms [3]. This makes them vulnerable to a wide range of cyber threats. Attackers can exploit vulnerabilities in IoT devices to gain unauthorized access, compromise data integrity, and disrupt critical services. Data privacy

[*]Computer Science and Engineering Department, National Institute of Technology Patna, India (himanshugbpuat@gmail.com,prabhat@nitp.ac.in).
[†]Computer Science and Engineering Department, Galgotias College of Engineering Technology, Greater Noida, India(kavitasharma$_0$6@*yahoo.co.in*).

is another critical concern. Many IoT devices collect sensitive information, including personal and location data. Unauthorized access to this data can lead to privacy breaches, identity theft, and legal consequences. Furthermore, the evolving cyber threat landscape poses a continuous challenge. Malicious actors are becoming increasingly sophisticated, using techniques like zero-day exploits and ransomware to target IoT vulnerabilities. To address these security challenges, effective solutions are essential and robust security measures need to be implemented.

Intrusion Detection Systems, also known as IDS, have traditionally been a primary line of defence against various types of cyberattacks [4]. The role of IDS in network security has been crystal clear: act as vigilant watchdogs, constantly monitoring traffic, detecting anomalies, and triggering alerts for potential threats. Traditional IDSs, built upon signature-based or rule-based mechanisms, have served well within the constraints of their design. However, the dynamism and complexity of IoT demand a more nuanced approach. Simple pattern matching or static rule sets are often ineffectual against sophisticated or zero-day attacks on IoT networks [6].

Over the past few years, machine learning and deep learning paradigms have emerged at the forefront of technological innovation and helped to cope up with such Security Concerns. Among the various architectures within deep learning, Recurrent Neural Networks (RNNs) hold particular promise for time-sequence data, which is intrinsic to network traffic in IoT. Unlike traditional feed-forward neural networks, RNNs possess the ability to 'remember' past inputs through their internal memory. This capability allows them to discern patterns in sequential data, making them particularly suited for IDS in IoT, where understanding temporal data sequences is crucial.

However, the mere existence of RNNs only sometimes translates to their effective implementation in IDS for IoT. Several challenges need to be addressed – the high dimensionality of network data, the real-time processing requirements of IoT, and the scalability concerns posed by billions of interconnected devices, to name a few. Moreover, while the application of RNNs in various domains like natural language processing or stock market prediction is well-documented, their tailored application for IoT intrusion detection is still nascent. This research seeks to bridge this knowledge gap, offering a comprehensive exploration of the design, implementation, and efficacy of an RNN-based IDS for IoT. As the narrative unfolds, the intricacies of the IoT landscape, highlighting its unique challenges, has been explained. Subsequently, an in-depth exploration of the RNN architecture will set the stage for understanding its applicability in the IDS domain. Through rigorous experimentation and evaluation, this research will not only propose but also validate the superiority of the RNN-based IDS for IoT, especially when compared against traditional models like J48, Random Forest (RF), Support Vector Machines (SVM), Multilayer Perceptron (MLP), and Naive Bayes(NB).

The core problem addressed in this research revolves around the inadequacy of existing intrusion detection and prevention systems to effectively safeguard IoT networks. Specifically, the research questions guiding this study include:

1. How can Recurrent Neural Networks (RNNs) be employed to detect and prevent intrusions in IoT networks?
2. What are the challenges and opportunities associated with implementing RNN-based techniques in the context of IoT network security?
3. How does the performance of RNN-based intrusion detection and prevention compare with traditional methods in terms of accuracy, adaptability, and real-time responsiveness?

The significance of this study lies in its potential to transform the landscape of IoT network security. By introducing a novel approach that leverages RNNs for intrusion detection and prevention, this research contributes to the development of adaptive and resilient security mechanisms for IoT networks. These mechanisms are vital to ensuring the continued growth and adoption of IoT technologies across various sectors, as security concerns have been a major impediment to realizing the full potential of IoT.

2. Literature Review.

This section examine the existing research on security and intrusion detection approaches for the Internet of Things.It will provide a comprehensive overview of the current state of IoT security, highlighting the vulnerabilities specific to IoT networks. Additionally, it will explore the various intrusion detection and prevention methods employed in traditional networks and IoT environments. This review will serve as the foundation for identifying gaps in the literature and setting the stage for the proposed RNN-based approach.

IoT applications are growing to smart grids, retail, residences, cities, and healthcare despite forecasts. Security is needed to avoid service disruption, illegal access, and cyberattacks like tampering and others to assure data accuracy and process efficiency. ML/DL is common in IDS development. IDSs protect IoT devices and systems from security and operation threats. Intrusion detection systems (IDS) are essential in IoT networks, which increasingly encompass critical infrastructure including healthcare, transportation, and energy. With the help of intrusion detection systems, network managers may quickly identify, address, and collect vital information needed to stop and lessen security risks.

Traditional machine learning methods like SVM [7], [8], K-Nearest Neighbor (KNN) [9], ANN [10], Random Forest (RF) [11], [12], and others [13] have been successful for intrusion detection systems. On the other hand, the DL method has outperformed ML in terms of accuracy, particularly for large datasets. Because picking features takes time and they won't know which characteristics are valuable until the model is trained and evaluated, researchers creating machine learning algorithms must exercise caution and only extract features that can improve the model. Machine learning is challenged when dealing with datasets of different sizes since it is not always easy to extract the most predictive features [14]. Furthermore, because deep learning models can independently extract properties from massive data sets, they outperform traditional machine learning techniques and are more accurate [15].

Kumari et al. proposed a semi-supervised intrusion detection system [16] using a hybrid SVM-FCM clustering platform for classification. This was an extra semi-supervised intrusion detection system. Active SVM uses a modest amount of labeled input and a lot of unlabeled data. This was done to prove that active learning SVM can identify like a typical support vector machine after N iterations. For multi-class classification, the FCM classifier was used on data items around support vectors. This model used SVM and FCM classifier engines for intrusion detection. If both classifiers regarded an input instance normal, we may confidently call it normal. If the SVM engine classified the input instance as an outlier and the FCM engine identified its sub-category, the instance is considered abnormal and the sub-class is selected by selecting the circle with the highest fuzzy membership and geographical proximity to the support vectors.

In order to identify malicious attacks in IoT contexts,Otoum et al. [17] introduced a novel DL-based intrusion detection system to resolve the challenges associated with protecting IoT nodes. In their proposed model, the spider monkey optimization (SMO) algorithm and the stacked deep polynomial network (SDPN) are combined to achieve the highest detection and recognition rates. SMO selects the most relevant attributes from the datasets, while SDPN classifies the output as normal or aberrant. Using DL to identify intrusions with recurrent neural networks (RNN-IDS) was recommended by Yang et al. [18]. They show through their experimental results that RNN-IDS is ideally adapted for producing IDS with good accuracy and that it outperforms conventional ML both binary and multi-class techniques. Dawoud et al. [19] presented a deep learning-based intrusion detection system for SDN-based IoT architecture. SDN modeling was used for the IoT security, scalability, and resilience enhancing purposes, whereas Restricted Boltzman Machine (RBM) was used as the engine for intrusion detection. This serves as an example of the integration of SDN and IoT. The suggested model was tested, evaluated, and validated by utilizing the KDD Cup'99 dataset, on which it produced a competitive performance more than 94% in terms of precision and accuracy.

Khan et al. [20] employed an ensemble-based voting classifier, where the final prediction was derived by combining the conventional machine learning algorithm with voting on its predictions. Using a stacking-based ensemble model, IoT devices are better able to detect anomalies in IoT networks, according to Naz et al. [21]. To enhance the effectiveness and precision of ensemble-based IDS,Bhati et al. [22] implemented ensemble-based IDS with XGBoost, which improves the accuracy. In [23], Arko et al. presented an overview of several machine learning techniques that can be used to identify potentially harmful or out-of-the-ordinary data, as well as the most effective approach for two datasets: the first dataset was created from data exchanged between sensors, and the second dataset is UNSW-NB15.

The use of ensemble learning, in which many methods/models or experts are put to use in order to solve a specific artificial intelligence-based problem, was another approach that researchers took in order to ensure the strong security of the IoT. In the context of the problem of intrusion detection, ensemble learning fosters stronger generalization, and the voting amongst the various strategies of ensemble give higher detection accuracy than the individual models, according to the proposal made by Illy et al. [24].

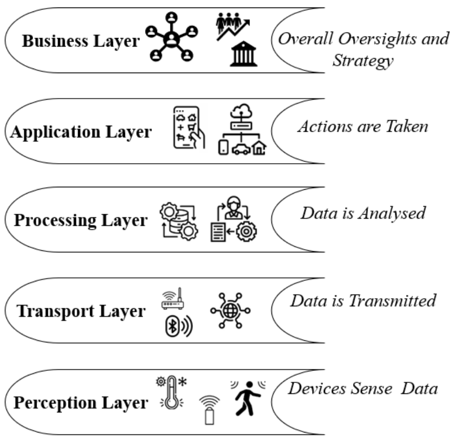

Fig. 3.1: IoT Architecture

In [25], Verma et al. investigated the viability of machine learning classification techniques for defending the IoT against DoS attacks. The Classifiers are evaluated using well-known datasets such as CIDDS-001, UNSWNB15, and NSL-KDD. Some cyber security experts have modified deep learning components to accomplish ML features for cyber-security, including IoT.

Yin et al. examined the structure of a deep learning-based intrusion detection system (IDS) and presents a novel RNN-IDS approach[26]. An comprehensive study examines the model's operationality in binary and multiclass classification scenarios and the effect of neuron count and learning rate changes on its efficacy. Using benchmark datasets, it is compared to J48, artificial neural network, random forest, and support vector machine. The authors suggest GPU acceleration to reduce training time, avoid exploding and vanishing gradients, and study the classification performance of LSTM, Bidirectional RNNs algorithms in intrusion detection.

Khan et al. proposed a deep learning-based intelligent IDS for IoT networks to address the security issues[27]. A Recurrent Neural Network with Gated Recurrent Units (RNN-GRU) can classify assaults across the physical, network, and application levels. This suggested model is trained and tested using the ToN-IoT dataset, which is unique for a three-layered IoT system and offers new attacks compared to other publicly available datasets. The proposed model's performance was analyzed using accuracy, precision, recall, and F1-measure, with Adam and Adamax optimization techniques. Adam was found to perform best.

3. IoT Architecture.

The Internet of Things, or IoT for short, is a bit like a huge, worldwide web where computers and everyday objects are connected to each other. Think of it as a world where your fridge, watch, car, and even your shoes can 'talk' to each other through the internet. For all these things to work smoothly, we need a plan or structure, just like building a house. This plan is called the IoT architecture. At its core, the IoT architecture can be described as multi-layered, each serving a specific purpose, working together to deliver an interconnected, intelligent ecosystem. Let's break down this architectural framework in Figure 3.1.

3.1. Perception Layer (Device Layer).

Imagine stepping into a dense forest, with every rustling leaf, chirping bird, or distant animal footstep communicating a piece of information. That's precisely the role of the Perception Layer. Often termed the physical or device layer, it's the frontline where real-world data is gathered. Comprising sensors, actuators, and other IoT devices, this layer perceives or senses the environment. Whether it's a smart thermostat sensing room temperature or an agricultural sensor gauging soil moisture, data collection begins here.

3.2. Transport Layer.

Having collected the data, the next step is its relay to central hubs for further action. Enter the Transport Layer. Acting as the communication bridge, this layer ensures data moves from devices to data centers using a myriad of transmission mediums[30]. This could be via satellite, cellular networks, Wi-Fi, or even more niche protocols like Zigbee. The fundamental task here is secure, swift, and efficient data transmission.

3.3. Processing Layer (Middleware Layer). All of the analysis that is done on data takes place at the Processing Layer. One could compare it to a location where information is stored, worked on, and interpreted. After being entered into databases, the raw data is subsequently transformed into information that can be utilized by specialized tools and computers. For example, by analyzing the data from a smart thermostat, it is possible to determine how to regulate the heating in order to save water and energy.

3.4. Application Layer. Application Layer is responsible to put the information received from Middle layer for practical use. Here, specific applications tailored for end-users interpret the processed data to offer tangible services. In a smart home setting, based on data from various sensors, the application layer might adjust lighting, heating, or even play your favourite song once you walk in. Essentially, this layer personalizes the IoT experience, translating processed data into relatable user actions[29].

3.5. Business Layer. Finally, at the top of all resides the Business Layer. Beyond the complexities of devices and data, this layer aligns the entire IoT architecture with overall business objectives. By analyzing data patterns, consumer behaviours, and device performance, strategic business decisions emerge. Whether it's launching a new product, optimizing an existing one, or even exploring uncharted market territories, this layer ensures the IoT system remains profitable, scalable, and aligned with overarching business objectives.

IoT architecture can be thought as a well-organized city where every part has a role. Every layer is crucial, from the devices that sense things to the pathways that transport data, the brains that process information, the hands that act on it, and the wise tree overseeing it all. This amazing plan lets our world of connected devices work together, making our lives easier and smarter. As more and more things around us start 'talking' to each other, knowing a bit about this architecture helps us appreciate the magic of the IoT world.

4. IDS for IoT. An Intrusion Detection System (IDS) for the Internet of Things (IoT) is paramount due to the inherent vulnerabilities associated with IoT devices and their increasing pervasiveness. Given the unique characteristics of IoT environments, traditional IDS might not be directly suitable, necessitating specialized approaches[7]. Here's a classification of Intrusion Detection Systems for IoT:

4.1. Based on Placement. Intrusion Detection Systems (IDS) can be fundamentally classified by their placement within the system they monitor. Host-based IDS (HIDS) operate on individual IoT devices. They focus on the internals of the device, such as system logs, processes, and system calls. Their primary advantage is their ability to effectively detect insider attacks and anomalies that manifest within a specific device. In contrast, Network-based IDS (NIDS) are centered on monitoring network traffic. By capturing and analyzing packets transmitted across the network, they are particularly apt at detecting unauthorized access or Distributed Denial of Service (DDoS) attacks that exploit network vulnerabilities[31].

4.2. Based on Detection Method. The detection methodology behind an IDS plays a critical role in its efficacy. Signature-based IDS operate using predefined patterns or signatures of known threats, making them adept at identifying recognized threats, but they are inherently limited when it comes to zero-day attacks. Anomaly-based IDS, on the other hand, rely on historical data to build a profile of what is considered "normal" behavior. When the current behavior deviates significantly from this profile, an alert is triggered. While this approach can detect previously unknown attacks, it might also lead to false positives. Specification-based IDS take a slightly different approach by using well-defined specifications that describe correct operation, and they raise alerts when there are deviations from these specifications. These are especially suitable for environments where correct behaviors can be meticulously defined. Lastly, Hybrid IDS merge the techniques of signature and anomaly-based detection to strike a balance between detection rates and false positives.

4.3. Based on the Type of IoT Environment. The type of IoT environment can dictate the design and priorities of an IDS. For instance, Home IoT IDS are designed specifically for smart home devices, such as thermostats, cameras, and smart appliances. They prioritize the privacy of users while ensuring usability. In industrial settings, the Industrial IoT (IIoT) IDS focus on ensuring system uptime, safety, and resilience in places like manufacturing plants and power grids. Healthcare IoT IDS, intended for medical devices and systems, place an unsurpassed emphasis on patient safety and data integrity. And in the realm of transportation,

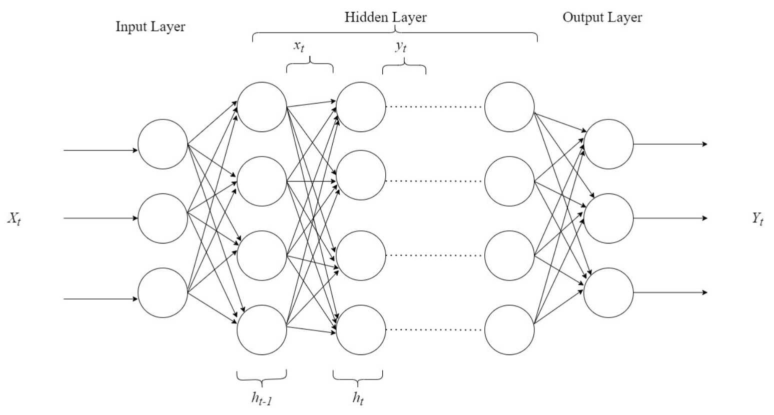

Fig. 5.1: RNN model

Vehicle IoT IDS cater to connected cars and vehicular networks, where passenger safety and real-time response are paramount.

4.4. Based on Operational Capability. In terms of operational capability, IDS can be collaborative or standalone in nature. Collaborative IDS are designed with multiple IDS nodes that collaborate and share information, offering a holistic view of the network and the ability to correlate events across diverse devices. This collaborative approach often leads to more robust detection and mitigation strategies. In contrast, Standalone IDS operate independently without the need for collaborative data. While they might be simpler and easier to deploy, they may lack the comprehensive view that collaborative systems offer. In essence, given the multifaceted nature of IoT devices and networks, an optimal IDS often necessitates a blend of these categorizations, each tailored to the unique requirements and threat landscapes of the environment.

5. Proposed Method. Proposed RNN-based IDS framework leverages the sequential processing capabilities of RNNs to analyze patterns in network traffic and detect intrusions. By using the NSL-KDD dataset, which is a benchmark in the IDS domain, the model can be trained to recognize a wide variety of intrusion patterns relevant to IoT environments. The combination of real-time processing, alert systems, and feedback loops ensures the IDS remains dynamic, effective, and up-to-date in the ever-evolving landscape of IoT security threats. Given the sequential nature of network traffic data, RNNs can potentially excel in identifying patterns and anomalies.

RNN has a looped or recurrent hidden layer, which allows it to maintain a 'memory' of previous inputs in its internal structure. This is what enables it to process sequences of data rather than single data points. As shown in FIG. 5.1, Inside Input layer, at each time step *t*, the RNN receives an input vector x$_t$. This vector will usually be an encoded form of the data for that time step, such as a word in a sentence or a feature in a time series.In Hidden Layer ,The recurrent layers computes the hidden state $_t$ at time step *t*. This hidden state is a function of the input x_t at the current time step and the hidden state h_{t-1} from the previous time step. In output layer, at each time step t, the RNN produces an output vector y$_t$. This output can be computed based on the hidden state h$_t$ and, if necessary, the input x$_t$.

Training an RNN involves adjusting its weights based on the difference between its predicted outputs and the actual outputs for a sequence. This is done using a variant of the backpropagation algorithm called Backpropagation Through Time (BPTT). BPTT works by unrolling the entire network for a sequence and applying the standard backpropagation algorithm, considering the temporal depth introduced by the recurrent layer.

As shown in FIG. 5.2 the process of an RNN-based Intrusion Detection System begins with the preprocessing of the Dataset. This includes label encoding, feature scaling, and feature selection to make sure the data is compatible. After the data has been pre processed, it is split into two sets: training (80%) and testing (20%).

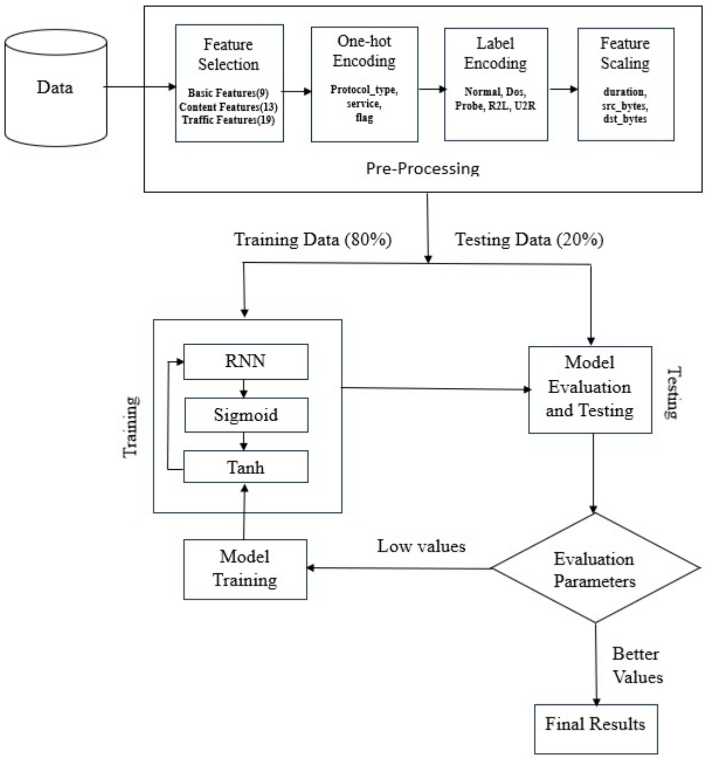

Fig. 5.2: Proposed RNN IDS

The training sample is used to build and train the RNN model.During training the model fits the data for 100 epochs with a batch size of 32. This is how it learns to spot patterns that are linked to intrusions. After that, the model is put to the test using the testing dataset to see how well it works using accuracy, precision, recall, and F1-score as measures. If the model's original evaluation metrics show that it isn't working as well as it could, it goes through incremental training, which involves fine-tuning its parameters even more until it gets good results. This iterative review process makes sure that the model's ability to find and stop cyber threats is always getting better. Once the model works well as it should, it can be used for real-time attack detection in live systems, which is a strong way to protect them.

5.1. Preprocessing of Dataset. Intrusion Detection Systems (IDS) often deal with large and complex datasets that require preprocessing to be effectively used for detecting malicious activities. Proper preprocessing is crucial as the quality and relevance of the data directly impact the model's performance. In this section various steps involved in Data Preprocessing are discussed followed by dataset description.

5.1.1. Dataset Description. NSL-KDD[28] is an improved version of the famous KDD Cup '99 dataset. The NSL-KDD dataset has made a name for itself in cybersecurity study, especially in the area of Intrusion Detection Systems (IDS). In this digital age, where network breaches and cyberattacks are getting smarter and happening more often, it is very important to have effective and accurate IDS. As a result, the NSL-KDD dataset has become an important tool for study into creating, testing, and improving different IDS models by providing a standard against which to measure and contrast their effectiveness. To fully understand what the NSL-KDD dataset is and how it can be used, it is important to go back to where it came from: the KDD Cup '99 dataset, which was created in 1999 as part of the Third International Knowledge Discovery and Data Mining Tools Competition. Even though it has problems like a huge number of duplicate records and built-in biases, the KDD'99 dataset quickly became the standard for IDS study. To address such limitations, the NSL-KDD was created to eliminate redundancies and give a more balanced dataset for constructing and assessing IDS models. NSL-KDD is notable for its comprehensive and diverse composition, encapsulating various aspects of network interactions and potential intrusions,this includes:

A. Variety of Features. It includes a large group of 41 features that cover a wide range of topics, such as basic features of each TCP connection, content features that show what's inside the packets, and traffic features that are estimated using a two-second time window. The features can be broadly divided into three groups namely Basic features ,Content features and Traffic features. Basic Features encompass attributes derived directly from the connection. Examples include the duration of the connection, the type of protocol used (e.g., TCP, UDP, ICMP), and other foundational data attributes. Content Features are derived from the content of the connections, such as the number of failed login attempts. These attributes provide insights into the suspicious behavior exhibited within the connection. Traffic Features are Computed with respect to a temporal window, these features capture network traffic statistics, analyzing patterns over a specified interval. These are further split into "time-based" and "connection-based" traffic features.

B. Multi-class Labels. Instances are divided into "normal" and several "attack" kinds. These are further broken down into four main types of attacks: DoS (Denial of Service), R2L (Remote to Local), U2R (User to Root), and Probing. Binary and Multi-class Classification: The attack types allow for both binary classification (normal vs. attack) and multi-class classification, which opens up a lot of theoretical and practical options.

C. Training and Test Sets. The dataset is split into "KDDTrain+" and "KDDTest+" sections, which make it easier to train, test, and validate models while keeping the lines between them clear to stop data leaks.

5.1.2. Feature Selection. In the realm of IDS for IoT, the relevance of features might differ from traditional network environments. For example, IoT devices often have resource constraints and unique patterns of network traffic. Therefore, selecting features that best characterize the IoT device behaviour is crucial. By focusing on the most relevant features, models can be more interpretable, faster, and potentially yield better performance The KDD'99 dataset initially has 41 features, categorized into basic features, content features, and traffic features.

Basic Features (9). These are derived from the packet headers without inspecting the payload, e.g., duration, protocol type, and service.

Content Features (13). These include features extracted from the payload like the number of failed login attempts.

Traffic Features (19). These are computed with respect to a window interval and are either time-based or connection-based. Using methods like Pearson's correlation coefficient can help determine if some features are highly correlated. If two features have high correlation, it might be beneficial to keep only one of them to avoid redundancy.

5.1.3. One-Hot-encoding:. One-Hot-Encoding is used to convert all categorical properties to binary properties. One-Hot-Endcoding requirement, the input to this transformer must be an integer matrix expressing values taken with categorical (discrete) properties. The output will be a sparse matrix in which each column corresponds to a possible value. It is assumed that the input properties have values in the range [0, n_values]. Therefore, to convert each category to a number, properties must first be converted with LabelEncoder.

There are 3 categorical attributes in this dataset are "Protocol_type", "service", and "flag" excluding "label" attribute. These features, although packed with essential information, are represented as text or categorical values, which are not inherently quantifiable and thus not directly compatible with RNN algorithms.

One-hot encoding is a favored technique for converting categorical data into a format that can be provided to RNN model. The process essentially creates a binary column for each category and indicates the presence of the category with a "1" or "0". Let's break down the one-hot encoding process for each of these features.

Protocol type. This feature indicates the type of protocol used for the connection, such as "tcp", "udp", or "icmp". Instead of these textual values, one-hot encoding would result in three new binary columns named "protocol_tcp", "protocol_udp", and "protocol_icmp". For a specific record in the dataset, if the protocol type is "tcp", the "protocol_tcp" column would have a value of "1" while the other two columns would be "0".

Service. The "service" attribute is a bit more complex as it delineates the network service on the destination, e.g., "http", "ftp", "telnet", and so on. Given the diverse range of services in the NSL-KDD dataset, one-hot encoding would result in multiple new binary columns, one for each service type. For instance, if a specific record has the service type ”http”, then the ”service_http” column would be ”1”, while all other ’service_*’ columns would be ”0”.

Flag. Representing the status of the connection, typical values might include ”SF”, ”S0”, ”REJ”, etc. Just like the earlier attributes, each unique flag value would get its binary column. So, if a specific connection record had its flag set as ”REJ”, the corresponding ”flag_REJ” column would hold a ”1”, with all other ’flag_*’ columns set to ”0”.

Post one-hot encoding, the NSL-KDD dataset will have an expanded feature set with new binary columns replacing the original categorical ones. This transformation ensures that the data is in a numerical format, making it suitable for RNN without losing the categorical information’s granularity. It’s crucial, however, to note that this process can increase the dimensionality of the dataset significantly, especially if categorical features have numerous unique values. As such, after one-hot encoding, dimensionality reduction techniques might be considered to optimize the dataset’s size without compromising the integrity of the information.

5.1.4. Label Encoding. The NSL-KDD dataset, a cornerstone in network intrusion detection research, underwent a transformation to simplify the representation of its diverse range of attacks. A large number of attacks were categorized into five broad categories, and to make these categories machine-friendly and facilitate easier computation, label encoding was applied. Initially, the dataset had various textual tags indicative of different kinds of attacks. To standardize and streamline this, the tags were remapped as follows:

Normal Activities: Previously labeled with various tags indicating normal behavior, these were consolidated and encoded with the value 0.

DoS (Denial of Service) Attacks: All tags specific to different types of DoS attacks(neptune, back, land, pod, smurf, teardrop, mailbomb, apache2, processtable, udpstorm, worm) were unified under the umbrella term ”DoS” and were encoded with the value 1.

Probe Attacks: These are attacks where the malicious actor scans the network to gather information or find known vulnerabilities. All such attacks(ipsweep, nmap, portsweep, satan, mscan, saint) were labeled as ”Probe” and assigned the encoded value 2.

R2L (Remote to Local) Attacks: In these attacks(ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, httptunnel), an attacker who does not have an account on the target machine tries to gain access. Such attempts, previously labelled with various specific tags, were brought together under ”R2L” and encoded with the value 3.

U2R (User to Root) Attacks: In U2R attacks, the attacker starts with access to a normal user account on the system and tries to exploit some vulnerability to gain root privileges. All such tags(buffer_overflow, loadmodule, perl, rootkit, ps, sqlattack, xterm) were encoded with the value 4.

The transformation process ensured the dataset became more streamlined. Instead of dealing with a multitude of tags that can make data processing and analysis cumbersome, especially for Deep learning models, we now have a standard set of five encoded labels. This not only helps in reducing the complexity but also in improving the efficiency of subsequent computations. To achieve this encoding, a straightforward mapping mechanism was used. A typical process would involve iterating over the dataset, examining the existing attack tag, and then replacing it with the new encoded value. This encoding, though seemingly simple, is a crucial step in data preprocessing, especially when the data is meant to be fed into machine learning or deep learning models. Properly encoded labels ensure models train effectively and provide meaningful results. Given the critical importance of network intrusion detection in today’s hyper-connected world, such streamlined data representations play a pivotal role in advancing cybersecurity research and solutions.

5.1.5. Feature Scaling. In the domain of data preprocessing for deep learning models, feature scaling stands as a pivotal step to standardize the range of independent variables or features of the data. This process is paramount, especially in datasets with features that have different scales, as it can drastically impact the performance of certain algorithms. The KDD dataset, renowned in the realm of network intrusion detection, is no exception to this rule. For the NSL-KDD dataset, taking into account the varying magnitudes, units, and range of the features, the decision was made to apply logarithmic scaling, a specialized scaling method. This method is particularly useful when dealing with data that spans several orders of magnitude. By applying logarithmic transformations, we can diminish the effects of outlier values and compress the scale on which the data lies, rendering it more manageable and interpretable. The features duration, src_bytes, and dst_bytes are taken into account. Their original ranges were considerably broad and spanned multiple magnitudes. However, by applying the logarithmic scaling method, these features were transformed to more condensed ranges. Specifically:

For the duration feature, post-logarithmic scaling, the range was condensed to [0, 4.77].

The src_bytes feature, after the application of logarithmic scaling, had its values fall within the range [0, 9.11].

Similarly, the dst_bytes feature was scaled such that its values now lie in the [0, 9.11] range.

It's noteworthy to mention that before applying the logarithmic scaling, a small constant might be added to the feature values to handle instances of values being zero, since the logarithm of zero is undefined. In essence, the logarithmic scaling of the NSL-KDD dataset's features ensures that the variances in the data's magnitude do not negatively influence the performance of machine learning algorithms. This transformation not only promotes better convergence during model training but also contributes to a more accurate and insightful representation of the underlying patterns and structures within the dataset.

5.2. Methodology. The purpose of this study was to use the NSL-KDD dataset to construct and assess an intrusion detection system (IDS) based on RNNs. Two different forms of configuration were used for proposed RNN-IDS: multiclass classification and binary classification. The goal of using both classification techniques was to evaluate proposed RNN model's adaptability and effectiveness in differentiating between different types of attacks and normal traffic.

An RNN-based intrusion detection system (IDS) is built and evaluated using the NSL-KDD dataset. The proposed RNN-IDS made use of two distinct configuration types: multiclass and binary classification. The purpose of combining the two classification methods was to test how well the suggested RNN model could distinguish between malicious and normal traffic.

Given an IoT network traffic dataset(NSL-KDD dataset), the task is to classify sequences of network data into one of N categories, such that "normal" and "Attack" in case of Binary Classification & "normal","Dos", "Probe", "R2L", and "U2R" in case of Multiclass Classification

Let:

X={x_1, x_2,..... x_T}: A sequence of feature vectors, where T is the length of the sequence.

Y={y_1, y_2,..... y_T}: The corresponding labels or categories.

The objective is to Model a Function f using RNN such that f(X) $\approx Y$

RNNs are designed to recognize patterns in sequences of data by utilizing memory elements. The primary component of the RNN is its hidden state, which gets updated at each time step of the sequence as depicted by Equation 5.1.

$$h_t \ = \ \sigma\left(Wx_t + Uh_{t-1} + b\right) \tag{5.1}$$

$$E = mc^2 \tag{5.2}$$

$$\int_a^b f(t)\left(\sum_i E_i B_{i,k,x}(t)\right) dt \tag{5.3}$$

where:

h_t is the hidden state at time t.

W and U are weight matrices

b is the bias vector.

σ is a non-linear activation function, often the hyperbolic tangent (tanh).

Output would be as follows:

$$y_t \ = \ \phi\left(Vh_t + c\right) \tag{5.4}$$

where:

V is the weight matrix for the output.

c is the output bias.

ϕ is a softmax function when the task is multi-class classification, providing a probability distribution over the N.

For training the IDS, a suitable loss function, L as depicted by equation 5.5, is categorical cross-entropy for classification tasks:

$$L(Y\ ,\ \hat{Y})\ = -\sum_{t=1}^{T}\sum_{n=1}^{N} y_{t,n}\ \log(\ \hat{y}_{t,n}) \tag{5.5}$$

where $\hat{y}_{t,n}$ is the predicted probability of the nth class at time t, and $y_{t,n}$ is a binary indicator (0 or 1) if label n is the correct classification for observation t.

For binary classification using RNN-IDS, the hidden layer uses 80 neurons and the activation function is hard sigmoid, while the output layer uses tanh. For instance, sensor updates are common examples of how IoT devices commonly broadcast data in a specified pattern. Anomaly detection may rely heavily on the temporal dependencies present in this data. On the other hand, a model based on RNNs could do better. A comparison is conducted to ascertain the RNN-IDS's effectiveness in relation to more conventional machine learning models. 'J48', 'RF', 'SVM', 'MLP', and 'NB' were used to assess RNN-IDS's performance by way of the evaluation metrics. In order to determine the F1 score, Precision, Accuracy, and Recall, these contrasting Machine Learning Models are constructed and implemented on the NSL-KDD dataset.

5.3. Evaluation. Using the NSL-KDD dataset, Proposed work used an RNN model to identify and classify four attack types: Dos, Probe, R2L, and U2R, along with normal traffic labels. The model's performance was carefully compared against five popular Machine Learning (ML) methods: J48, Random Forest (RF), Support Vector Machines (SVM), Multilayer Perceptron (MLP), and Naive Bayes(NB).

To allow for thorough comparisons, the model was trained on a stratified dataset split and evaluated using the following metrics.

Accuracy: The percentage of true predictions in our model compared to total predictions.

Precision: It measures positive prediction accuracy, calculated as the ratio of true positive outcomes to the sum of true positives and false positives.

Recall (Sensitivity): Rate of true positive predictions compared to true positives and false negatives.

F1 Score: It is a balanced measure of precision and recall, particularly in skewed datasets. Can be calculated as the harmonic mean of Precision and Recall.

The performance of the RNN-IDS model for binary classification (Normal, anomaly) and multi class classification (such as Normal, DoS, R2L, U2R, and Probe) has been the subject of investigation in two separate studies that have been created specifically for this purpose. These experiments are designed at the same time as standard experiments and results are compared other machine learning strategies J48, NB, RF, MLP, SVM.

A variety of errors occurred during the development of the RNN-based model for the binary and multiclass categorization of different attacks, each providing information about a different set of difficulties. Overly sensitive models or data noise can cause False Positives (FP), in which the model incorrectly predicts an attack. This is avoided by modifying the threshold of the model and using post-processing. Conversely, attacks that the model was unable to identify as False Negatives (FN) were caused by over-regularization and inadequate modeling of specific attack patterns. Reducing FN errors required improving feature representation, taking into account more intricate models, or modifying regularization. Accurate forecasts of attacks are indicated by True Positives (TP) and True Negatives (TN), respectively. Constant fine-tuning, feature engineering, and striking a balance between sensitivity and specificity is used to improve model accuracy. Errors resulted from missing and irrelevant features, hence feature representation is regularly assessed and improved.

5.3.1. Binary Classification. In the binary classification scenario, the NSL-KDD dataset was framed to classify network activities into two broad categories: 'Normal' and 'Attack'. The 'Attack' category encompassed all four varieties of attack labels (Dos, Probe, R2L, and U2R), consolidating them into a single overarching class and making a binary classification setup possible. LSTM input layer must be 3D the meaning of the 3 input dimensions are: samples, time steps, and features. The number of samples is assumed to be 1 or more.

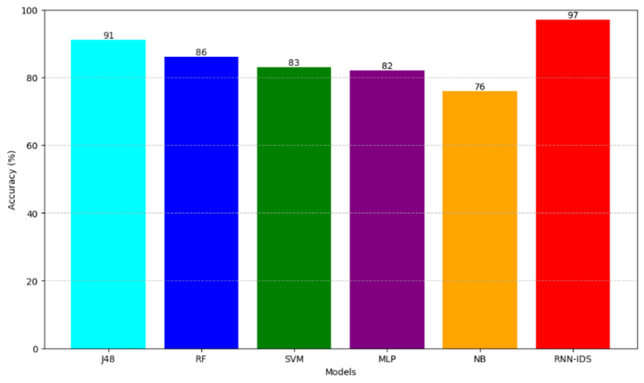

Fig. 5.3: Comparison of Accuracy between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset.

reshape() function takes a tuple as an argument that defines the new shape. To obtain strong comparative analysis with other models, the model was trained on a NSL-KDD dataset and evaluated using the following metrics.

Accuracy. Accuracy is calculated as the proportion of correctly predicted instances to the total number of instances in the dataset. It is a metric used to evaluate the IDS model's ability to correctly classify network traffic as either normal or malicious. Mathematically, accuracy can be expressed as follows:

$$Accuracy = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances (P + N)}} \tag{5.6}$$

True Positives (TP) is the count of attack instances correctly identified as attacks.

True Negatives (TN)True Negatives (TN) is the count of normal instances correctly identified as normal.

P is the total actual positive instances (actual attacks).

N is the total actual negative instances (actual normal activities).

The performance of RNN-IDS model in terms of Accuracy is superior to other classification algorithms in binary classification as shown in FIG. 5.3.

For Binary classification Accuracy of RNN-IDS came out to be 97% which is 6% more as compared to the accuracy of best model among as compared with other.

Precision. Precision, also known as the positive predictive value, is an essential evaluation metric, where the cost of false positives (incorrectly identified as an attack) may be significant. Precision attempts to assess the accuracy of the IDS, i.e., how many instances classified as positive (attack) are in fact positive.

Mathematically, precision is calculated using the following formula:

$$Precision = \frac{\text{TruePositives(TP)}}{\text{TruePositives (TP) +FalsePositives(FP)}} \tag{5.7}$$

where:

True Positives (TP) is the number of attack instances that were correctly identified as attacks.

False Positives (FP) is the number of normal instances that were incorrectly identified as attacks.

From FIG. 5.4 it can be observed that the Proposed RNN-IDS achieved a precision score of 0.95 which is a way better than other compared models during Binary Classification. This highlights the system's capacity to reduce false positives and provides the percentage of true positive predictions among all positive predictions.

Recall. Recall, also known as Sensitivity or True Positive Rate, is a crucial evaluation metric, revealing the model's ability to correctly identify and classify positive (attack) instances. It resolves the question: " How many true positive instances did the model successfully identify as being positive?".

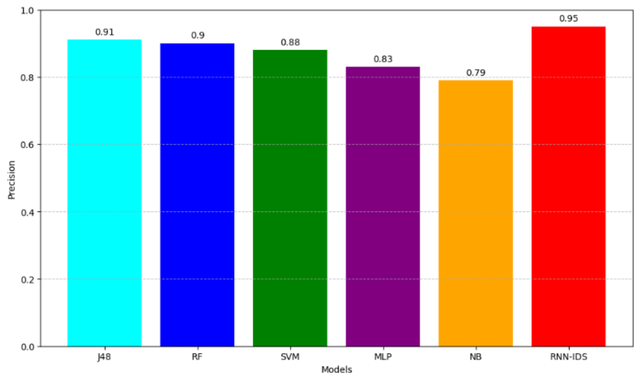

Fig. 5.4: Comparison of Precision between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset.

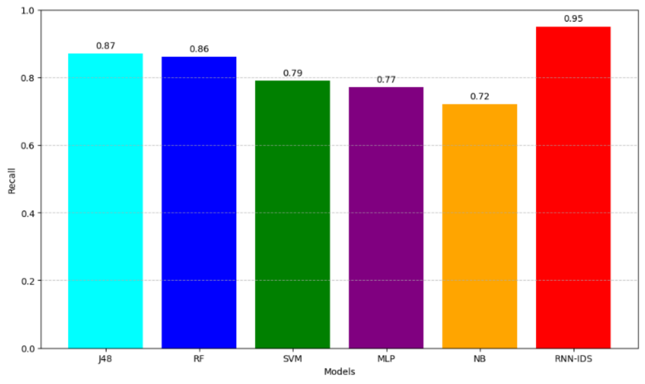

Fig. 5.5: Comparison of Recall between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset.

Mathematically, recall is computed as follows:

$$\mathit{Recall} = \frac{\text{True Positives(TP)}}{\text{True Positives (TP) +False Negatives(FN)}} \tag{5.8}$$

where:

– True Positives (TP): Represent the instances which were attacks and were correctly identified as attacks by the IDS.

– False Negatives (FN): Represent the instances which were attacks but were incorrectly identified as normal by the IDS.

Proposed RNN-IDS exhibited a recall score of 0.95, which represents the ratio of true positive predictions to all actual positive instances, underlining the system's capacity to identify actual intrusions effectively. It can be clearly depicted from the FIG. 5.5 that Proposed RNN-IDS surpassed all the other compared models in terms of Recall.

Score. The F1 Score is the harmonic mean of precision and recall, and it offers a balance between the two factors whenever there is an imbalance in the class distribution. It takes into consideration both false positives and false negatives, and it is especially useful in circumstances in which one form of error is more substantial than the other.

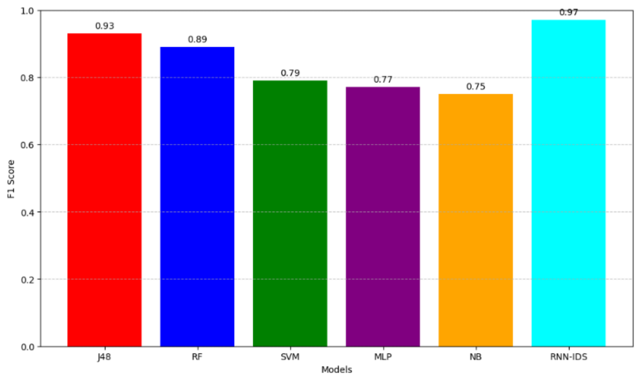

Fig. 5.6: Comparison of F1 Score between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset.

Mathematically, the F1 Score is defined as:

$$F1 - Score_{=}\ 2 * \frac{\text{Precision } * \text{ Recall}}{\text{Precision} + \text{Recall}} \tag{5.9}$$

where:

Precision (Positive Predictive Value) is defined as:

Precision $= \frac{\text{True Positives(TP)}}{\text{True Positives(TP)+False Positives(FP)}}$

Recall (Sensitivity or Tue Positive Rate) is defined as:

Recall $= \frac{\text{True Positives(TP)}}{\text{True Positives(TP)+False Negatives(FN)}}$

The F1 score, harmonizing precision and recall, for proposed RNN-IDS was 0.97. FIG. 5.6 reflects the model's performance in binary classification in terms of F1-score as compared to other models.

5.3.2. Multi Class Classification. The dataset was organized using the multiclass classification paradigm to categorize network activity into five different labels: ”Normal” and four classes of attacks (Dos, Probe, R2L, and U2R). The RNN model was put through a rigorous training program before being put to the test to see how well it could classify data across these many classifications.

Accuracy. In a multi-class classification problem with more than two classes, like four types of attacks and one normal label, the formulation might get a little more complicated because we have to figure out the True Positives and True Negatives for each class separately and then add them all up. If this is the case and C is the number of classes, the accuracy may be represented as:

$$Accuracy_{=} \frac{\sum_{i=1}^{C} \mathrm{TP_i} + \mathrm{TN_i}}{\text{Total Instances}} \tag{5.10}$$

TP_i and TN_i refer to the True Positives and True Negatives for the ith class respectively.

Proposed RNN-IDS recorded an accuracy of 95%.It can be observed in FIG. 5.7, when compared to the machine learning models, the RNN's performance was superior, indicating its adeptness in correctly classifying instances.

Precision. When dealing with multiple attack types (classes) in an IDS for IoT multi-class classification scenario, the precision for each class is calculated separately and then the macro-average precision is derived across all classes. This gives a general idea of the IDS model's precision across various attack types.

In a multi-class context, the following formula can be modified to compute class-wise precision as:

$$Precision_i = \frac{\mathrm{TP_i}}{\mathrm{TP_i} + \mathrm{FP_i}} \tag{5.11}$$

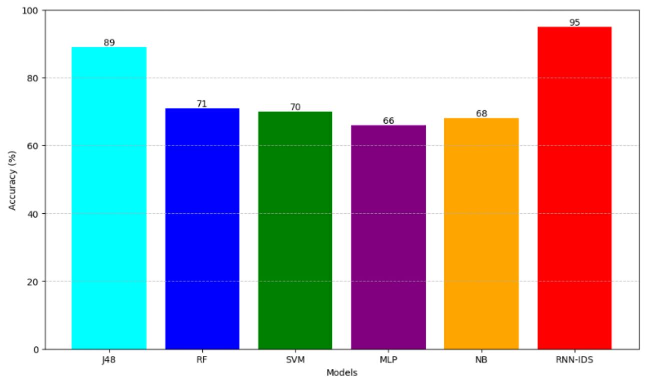

Fig. 5.7: Comparison of Accuracy between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset for Multiclass Classification.

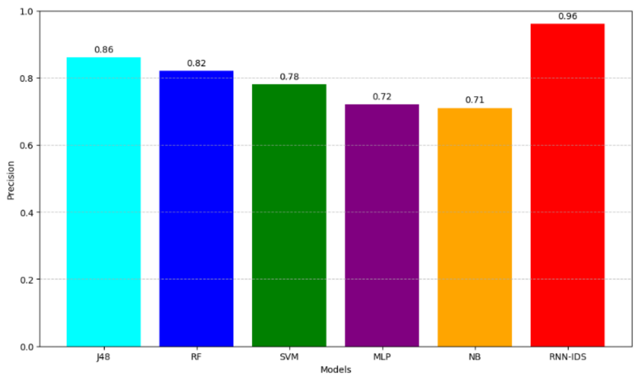

Fig. 5.8: Comparison of Precision between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset for Multiclass Classification.

where:

– Precision$_i$ is the precision for the ith class (type of attack).

– TPi and FPi represent the True Positives and False Positives for the ith class respectively.

After calculating the precision for each class, the macro-average precision across all classes is computed as follows:

$$Macro - AveragePrecision = \frac{\sum_{i=1}^{C} \mathrm{Precision_i}}{C} \tag{5.12}$$

Here C is the total number of classes.

Precision indicates the system's capacity to reduce false alarms, which is essential for IoT IDS usability and reliability. This metric is examined alongside recall and F1 score to evaluate the proposed model.

Precision is vital as it tells us about the model's capability to correctly identify positive instances. With a precision score of 0.96, the RNN-IDS model edged out most ML-based models, showcasing its reliability in positive identifications in FIG. 5.8.

Recall. In multi-class classification for IDS in IoT, there are several sorts of attacks (classes), recall has been computed for each class and then the macro-average recall is calculated over all classes to provide a generalized model recall measure. Class-wise recall for multi-class classification can be computed as:

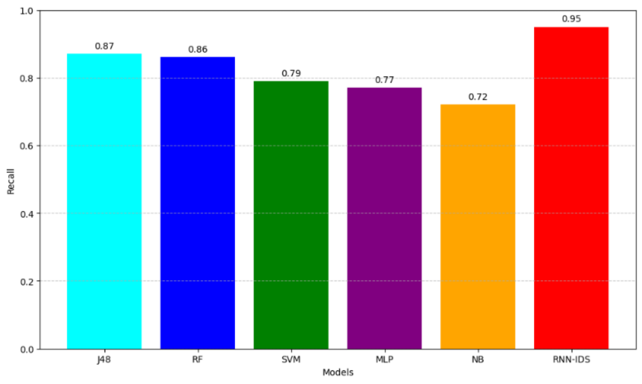

Fig. 5.9: Comparison of Recall between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset for Multiclass Classification.

$$Recall_i \; = \frac{\mathrm{TP_i}}{\mathrm{TP_i} + \mathrm{FN_i}} \tag{5.13}$$

where:

– Recall$_i$ is the recall for the ith class (type of attack).

– TP$_i$ and FN$_i$ are the True Positives and False Negatives for the ith class, respectively. Macro-average recall across all classes in a multi-class classification scenario can be derived as follows:

$$Macro - AverageRecall = \frac{\sum_{i=1}^{C} \mathrm{Recall_i}}{C} \tag{5.14}$$

where C is the total number of classes.

IDS for IoT relies on recall because missing an attack can be disastrous. This comprehensive evaluation helps fine-tune the model for reliable IoT IDS.

Recall focuses on the model's ability to identify all potential positive instances. The RNN-IDS model achieved a commendable recall score of 0.95, which was notably higher than some ML models as can be seen in Fig. 5.9, emphasizing its proficiency in identifying actual attack instances.

F1 Score. In a multiclass classification scenario, such as categorizing various types of network intrusions in IDS for IoT, the F1 Score can be calculated for each class separately, and then an average can be calculated to evaluate the classifier's overall performance. Micro and macro F1 Scores are two ways to calculate the average F1 Score for multiclass classification problems.

Micro F1 Score: Calculated by aggregating the contributions of all classes to find the average.

$$F_{1micro} = 2x\frac{\sum_{i=1}^{C} \mathrm{TP_i}}{\sum_{i=1}^{C} \mathrm{TP_i} + \sum_{i=1}^{C} \mathrm{FP_i} + \sum_{i=1}^{C} \mathrm{FN_i}} \tag{5.15}$$

where C represents the number of classes, and TP$_i$,FP$_i$, and FN$_i$ denote the true positives, false positives, and false negatives for the i-th class, respectively.

Macro F1 Score: The arithmetic mean of the per-class F1 Scores.

$$F_{1macro} = \frac{1}{C}\sum_{i=1}^{C} \mathrm{F1}_i \tag{5.16}$$

where F1$_i$ represents the F1 Score for the i-th class.

The F1-Score serves as a balanced measure, taking into account both precision and recall. Proposed RNN-IDS model's score of 0.94 was demonstrably superior, revealing its balanced performance in precision and sensitivity,as can be depicted in FIG. 5.10.

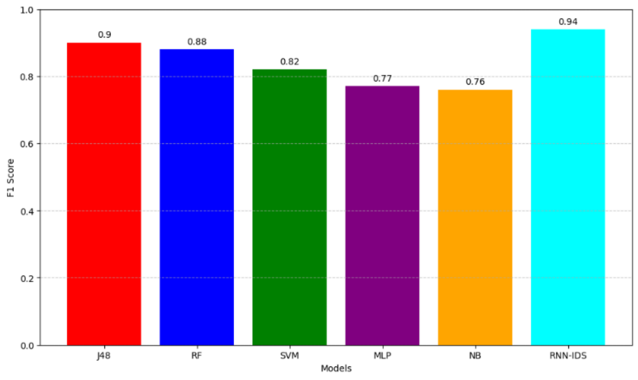

Fig. 5.10: Comparison of F1 Score between Proposed(RNN-IDS) and other(j48,RF,SVM,MLP,NB) methods applied on NSL-KDD dataset for Multiclass Classification.

6. Conclusion. The results of experiment demonstrate the effectiveness of proposed RNN-IDS for intrusion detection in both binary and multiclass classification scenarios. In binary classification, proposed RNN-IDS system excelled with high accuracy, precision, recall, and F1 score, indicating its ability to accurately identify both normal and intrusive network activities while minimizing false alarms.Furthermore, in the multiclass classification setting, proposed RNN-IDS showcased its adaptability by accurately classifying various intrusion types. This capability is crucial for network administrators and security professionals, as it enables them to pinpoint specific attack categories for prompt mitigation.

Comparing proposed RNN-IDS with renowned machine learning models: J48, RF (Random Forest), SVM (Support Vector Machine), MLP (Multi-Layer Perceptron), and NB (Naive Bayes), deduced that it consistently outperformed them in terms of accuracy, precision, recall, and F1 score. This suggests that the use of recurrent neural networks offers substantial advantages over conventional techniques when it comes to intrusion detection on the NSL-KDD dataset.

Proposed research proves the potential of RNN-based IDS systems in enhancing network security. The results indicate that proposed RNN-IDS is a promising approach for accurately detecting network intrusions, and its superior performance over traditional models makes it a valuable asset for real-world cyber security applications.While the current research demonstrates the effectiveness of RNN based intrusion detection systems, certain limitations highlight avenues for future exploration. Expanding the research to include larger and more diverse datasets may improve the model's resilience and generalization. In addition, the current work focuses on simulated environments, therefore deploying the model in real-world IoT settings would provide useful insights into its practical efficacy.

REFERENCES

[1] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, ”Network intrusion detection for IoT security based on learning techniques,” *IEEE Communications Surveys & Tutorials*, 21(3), (2019), pp. 2671-2701.

[2] S. Bajpai and K.S.B.K. Chaurasia, ”Intrusion detection system in IoT network using ML,” *NeuroQuantology*, 20(13), (2022), pp. 3597.

[3] A. Sinha, G. Shrivastava, and P. Kumar, ”Architecting user-centric internet of things for smart agriculture,” *Sustainable Computing: Informatics and Systems*, 23, (2019), pp. 88-102.

[4] M.J.S. Aneja, T. Bhatia, G. Sharma, and G. Shrivastava, ”Artificial intelligence based intrusion detection system to detect flooding attack in VANETs,” In *Handbook of Research on Network Forensics and Analysis Techniques*, IGI Global, (2018), pp. 87-100.

[5] Y. Otoum and A. Nayak, ”As-ids: Anomaly and signature based ids for the internet of things,” *Journal of Network and Systems Management*, 29(3), (2021), pp. 23.

[6] D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R.M. Mohammad, ”Intrusion detection system using feature extraction with machine learning algorithms in IoT,” *Journal of Sensor and Actuator Networks*, 12(2), (2023), pp. 29.

[7] H. Sharma, P. Kumar, and K. Sharma, ”Identification of Device Type Using Transformers in Heterogeneous Internet of

Things Traffic," International Conference On Innovative Computing And Communication, Singapore: Springer Nature Singapore.,(2023), pp. 471-481.

[8] R.R. Reddy, Y. Ramadevi, and K.N. Sunitha, "Effective discriminant function for intrusion detection using SVM," 2016 International conference on advances in computing, communications and informatics (ICACCI), (2016), pp. 1148-1153. IEEE.

[9] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on KNN classification algorithm in wireless sensor network," *Journal of Electrical and Computer Engineering*, (2014).

[10] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," 2015 international conference on signal processing and communication engineering systems, (2015), pp. 92-96. IEEE.

[11] N. Farnaaz and M.A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, 89, (2016), pp. 213-217.

[12] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl.Rev.*, vol. 38, no. 5, (2008), pp. 649-659.

[13] J.A. Khan and N. Jain, "A survey on intrusion detection systems and classification techniques," *Int. J. Sci. Res. Sci., Eng. Technol.*, vol. 2, no. 5, (2016), pp. 202-208.

[14] S.A. Bini, "Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care?," *The Journal of arthroplasty*, 33(8), (2018), pp. 2358-2361.

[15] X. Wang and X. Lu, "A host-based anomaly detection framework using XGBoost and LSTM for IoT devices," *Wireless Communications and Mobile Computing*, (2020), pp. 1-13.

[16] V.V. Kumari and P.R.K. Varma, "A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), (2017), pp. 481-485. IEEE.

[17] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: a deep learning–based intrusion detection framework for securing IoT," *Transactions on Emerging Telecommunications Technologies*, 33(3), (2022), p. e3803.

[18] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet of things Journal*, 4(5), (2017), pp. 1250-1258.

[19] A. Dawoud, S. Shahristani, C. Raun, "Deep learning and software-defined networks: towards secure iot architecture," *In. Things.*, 3–4, (2018), pp. 82-89.

[20] M. Khan, M. Khattk, S. Latif, A. Shah, M. Ur Rehman, W. Boulila, et al., "Voting classifier-based intrusion detection for IoT networks," in *Advances on Smart and Soft Computing*, Springer, (2022), pp. 313–328.

[21] N. Naz, M. Khan, S. Alsuhibany, M. Diyan, Z. Tan, M. Khan, et al., "Ensemble learning-based IDS for sensors telemetry data in IoT networks," *Math. Biosci. Eng.*, 19, (2022), pp. 10550–10580.

[22] B.S. Bhati, G. Chugh, F. Al-Turjman, and N.S. Bhati, "An improved ensemble based intrusion detection technique using XGBoost," *Transactions on emerging telecommunications technologies*, 32(6), (2021), pp. e4076.

[23] A. Arko, S. Khan, A. Preety, M. Biswas, "Anomaly Detection In IoT Using Machine Learning Algorithms," Brac University, (2019).

[24] P. Illy, G. Kaddoum, C.M. Moreira, K. Kaur, S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," (2019), pp. 15–18.

[25] A. Verma, V. Ranga, "Machine Learning intrusion detection systems for IoT applications," *Wireless Pers. Commun.*, 111, (2020), pp. 2287–2310.

[26] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, 5, (2017).

[27] Noor Wali Khan, Mohammed S. Alshehri, Muazzam A. Khan, Sultan Almakdi, Naghmeh Moradpoor, Abdulwahab Alazeb, Safi Ullah, Naila Naz, and Jawad Ahmad, "A hybrid deep learning-based intrusion detection system for IoT networks," *Mathematical Biosciences and Engineering*, 20(8), (2023), pp. 13491-13520.

[28] R. Zhao, "NSL-KDD," *IEEE Dataport*, (2022).

[29] A. Sinha, P. Kumar, N.P. Rana, R. Islam, and Y.K. Dwivedi, "Impact of internet of things (IoT) in disaster management: a task-technology fit perspective," *Annals of Operations Research*, 283, (2019), pp. 759-794.

[30] A. Arora, S.K. Yadav, and K. Sharma, "Denial-of-service (dos) attack and botnet: Network analysis, research tactics, and mitigation," In *Research Anthology on Combating Denial-of-Service Attacks*, (2021), pp. 49-73. IGI Global.

[31] S.K. Yadav, K. Sharma, and A. Arora, "Security integration in ddos attack mitigation using access control lists," *International Journal of Information System Modeling and Design (IJISMD)*, 9(1), (2018), pp. 56-76.

Edited by: Himani Bansal
Special issue on: Recent Advance Secure Solutions for Network in Scalable Computing
Received: Dec 8, 2024
Accepted: Apr 28, 2024