Deep Reinforcement Learning based Network Intrusion Classification

By

Khorshed Alam, ID: 012 231 0004

Submitted in partial fulfilment of the requirements
of the degree of Master of Science in Computer Science and Engineering

December 11, 2024

Department of Computer Science and Engineering
United International University

Abstract

Network intrusion classification referred to the process of monitoring and analyzing network traffic to identify suspicious activities or attacks. In this work, author proposed a novel approach to classify network intrusion by utilizing deep reinforcement learning (DRL), integrating a hybrid deep learning model architecture that combined Deep Neural Network (DNN) and Long Short-Term Memory (LSTM) networks. A DRL-based approach improved upon traditional deep learning by adapting dynamically to novel/unknown and evolving attack patterns. Unlike static models, DRL continuously learned optimal strategies through interaction with the environment, allowing for better detection of previously unseen threats in real-time. To address the class imbalance often encountered in network intrusion datasets, I evaluated the performance of several advanced data balancing techniques, including Borderline-SMOTE, SMOTE-ENN, ADYSN, and K-means SMOTE. The findings demonstrated that the K-means-based data balancing method outperformed other techniques, resulting in the most robust performance across various metrics. Author conducted multi-dataset validation to ensure robustness across different network flow data. For adaptive modeling testing, author excluded some attack types from training data and included them in testing data (e.g., DoS attacks were excluded from the training data but included in the testing data). The proposed approach enhanced the accuracy and reliability of intrusion detection, making it a viable solution for securing modern network infrastructures.

Acknowledgements

This work would have not been possible without the input and support of many people over the last two trimesters. I would like to express my gratitude to everyone who contributed to it in some way or other.

My sincere gratitude goes to my project Supervisor, Prof. Dr. Dewan Md. Farid. Without his constant support, pulling off this work would have been impossible.

I am also thankful to the Head Examiner and Reviewer, Prof. Dr. Al-Sakib Khan Pathan. He provided some valuable comments in the pre-defense which helped a lot to shape the final version of the work.

I would also like to thank Prof. Dr. Md. Motaharul Islam, Director, MSCSE, United International University and Professor Dr. Mohammad Nurul Huda, Departmental Head of Computer Science and Engineering (CSE), United International University for their invaluable support.

Last but not the least, I owe to my parents for their unconditional love and immense emotional support.

Publication List

The main contributions of this research are either published or accepted or in preparation in journal and conference as mentioned in the following list:

Journal Article

1. Khorshed Alam, Mahbubul Haq Bhuiyan and Dewan Md. Farid. Adaptive Network Intrusion Classification : A Deep Reinforcement Learning Approach (Submitted in IEEE Access Journal)

Conference Paper

1. M. H. Bhuiyan, K. Alam, K. I. Shahin and D. M. Farid, "A Deep Learning Approach for Network Intrusion Classification," 2024 IEEE Region 10 Symposium (TENSYMP), New Delhi, India, 2024, pp. 1-6, doi: 10.1109/TENSYMP61132.2024.10752251.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Network Intrusion Classification (NIC)

Network Intrusion Classification (NIC) plays a vital role in protecting computer networks by monitoring network traffic for signs of suspicious activity, such as malware, unauthorized access, or intrusion attempts [1]. These systems analyze incoming and outgoing data in real-time, looking for anomalies or patterns that indicate potential threats. It works much like an early warning system to the network administrators against potential attacks, and appropriate responses can be done much before considerable damage is caused. Due to continuously evolving advancement of the cyber threats, the development of a perfect NID is of prime importance. Attackers continuously vary their techniques to get past traditional security measures like firewalls and antivirus programs. Ideally, the NID would do this not only for known attack patterns with high efficiency but also for novel threats never seen before with high accuracy. Besides, a faultless NID would reduce false negatives-when benign activities are flagged as attacks-which becomes very important in keeping unnecessary interventions at their minimum and ensuring no real threats slip by. Since it emerges more sophisticated and resilient, the better that NID can protect integrity, confidentiality, and availability of network resources, its development now forms one of the cornerstones in modern cybersecurity strategies.

Research in computer networking becomes vital because it forms the backbone of modern communication that runs everything from personal use of the internet to worldwide business operations. The complexity and scale of networks keep on increasing as newer technologies such as 5G, cloud computing, and IoT are stretching traditional infrastructure beyond its limits [2]. AI involvement in networking transforms how the networks will be managed, optimized, and kept secure. AI-powered solutions [3] enable intelligent traffic routing, predictive maintenance, and real-time intrusion detection, ensuring networks are resilient, efficient, and adaptive to dynamic conditions [4]. As network infrastructures evolve, ongoing research is essential to address challenges like scalability, security, and performance optimization, making it pivotal for both technological advancement and the seamless functioning of digital economies.

1.2 Deep Learning and Deep Reinforcement Learning Based in NIC

In the area of NIC, two types of methodologies have been identified in the previous study [5, 6]. They are the DL-based methodology and the DRL-based methodology [7], with each having merit order based on design and application in use. This methodology from DL-based uses neural networks for processing data volumes that arise in network traffic. This will cause the system to detect and extract complex patterns and features. As the power of this technique rests in classification, it provides an easy pathway for DL-based NID to catch recurring or established attack patterns, considering their training would deal with high-quality substantial datasets of past attack knowledge. Thus, it is more appropriate and helpful in environments where high intrusion detection accuracy of known intrusions is required. While providing all-encompassing historical attack data, a DL-based NID will work just fine; these models become very good at finding recurring threats and well-documented attack signatures.

The DRL-based approach introduces an element of adaptability and continuous learning into intrusion detection. Unlike DL, which operates on static datasets, DRL involves an agent interacting with the network environment, receiving real-time feedback, and adjusting the detection strategy in pursuit of the same. This allows DRL-based systems to adapt dynamically to variations in network conditions and to the emergence of new or unknown threats. The adaptability thus provided makes a DRL-based approach particularly suitable for environments where novel or evolving threats are common-for instance, in government agencies or financial institutions, where attackers frequently employ either custom or advanced attack methods. This ability of the DRL model to learn and readjust in real time gives it an edge in handling sophisticated and unpredictable intrusion attempts. DL-based NIC applies to organizations that require high accuracy on known threats by using historical data, whereas DRL-based NID applies in scenarios where organizations require systems that are flexible and adaptive in handling emerging threats differently. Both approaches boast unique strengths, and the choice depends on the specifics of each organization's requirements and the kind of threat it faces. [7]

1.3 Novel Threat Detection

The DRL-based approach in NID is efficient in the detection of zero-day attacks due to its capability of continuous learning from the network environment [8]. In a typical DRL framework, there is an agent that perceives network traffic as a state and takes an action whether the observed traffic is a threat or not. In time, the agent receives feedback in terms of rewards or penalties depending on whether decisions taken are correct or not. This learning can enable the agent to dynamically refine the detection strategies and enhance the capability of detecting new patterns and anomalies in network flow that might indicate previously unknown attack techniques. While static models rely on preprogrammed rules

and patterns, DRL systems can adapt in real time and may therefore detect sophisticated and changing cyber-attacks. Among the key advantages of the adaptive modeling features of DRL are that the system becomes enabled to continuously optimize intrusion detection capabilities when its exposure to new types of attacks increases. Most of the traditional systems require updates manually if some new type of attack is identified. In contrast, this is where DRL-based NID can excel: with its self-adapting behaviors and continuous learning from its environment, it can perform well in zero-day attacks or other evolving threats. Because of behaviors that differ from normal ones even though those behaviors were at first not recognized in its training set-the DRL model can identify the anomalies within the network data, not bound by static signatures or any predefined knowledge. This adaptability makes DRL-based NID very effective in maintaining security under constant changes within the threat landscape. The DRL-based systems continuously learn and adapt; therefore, they form a robust defense mechanism against known and unknown threats, bringing in proactive and resilient network security.

1.4 Motivation

The motivation for using DRL in the network intrusion classification problem is because cyber threats are getting increasingly sophisticated and adaptable, which sometimes are not capable of being dealt with by a conventional IDS. Traditional methods, such as signature-based or rule-based systems, depend on predefined patterns and are usually static; thus, they easily become susceptible to novel evolving zero-day attacks. While DRL will provide an adaptive intrusion detection method wherein systems learn continuously through real-time interaction with the network environment. In addition, having the feedback mechanism follow naturally whereby the system gets rewarded or penalized for its actions will make the model iteratively refine its detection strategy toward emergent threats and anomalies. This adaptability ensures DRL-based systems will not be bound to predefined attack signatures but can generalize to detect previously unseen intrusion patterns. Besides, integrating DRL in network intrusion classification blends with modern complex network environments-like ones propelled by IoT, cloud computing, and 5G-technologies where the attack vectors change at high speed. The fact that DRL can balance high detection accuracy with minimal false positives, while at the same time adapting dynamically to ever-changing cyber threats, makes it a formidable choice for next-generation intrusion detection systems. This guarantees robust, proactive security in networking.

1.5 Contributions

Author proposed a DRL based Network intrusion classification framework powered by DNN+LSTM hybrid deep learning architecture, which can detect novel attack variants of network flow data. The contribution of our work is stated below:

- Proposed a novel approach to network intrusion classification using deep reinforcement learning (DRL), integrating Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) networks.
- Enhanced real-time detection of previously unseen threats, improving upon the limitations of static models in traditional deep learning approaches.
- Addressed the class imbalance issue prevalent in network intrusion datasets by evaluating the performance of various advanced data balancing techniques: Borderline-SMOTE, SMOTE-ENN, ADYSN, and K-means SMOTE.
- Author has noticed many DRL-based approach faces difficulties while detecting slow-paced attacks discussed in chapter III. In order to make the model more robust for detecting slow paced attacks we have included new features in dataset which contributes to the accurate classification discussed in chapter IV-B.
- Author has conducted experiment on adaptive modeling functionality of our model which showed good performance against unseen/novel attacks discussed in chapter VI.

1.6 Thesis organization

In chapter I, the introduction emphasizes the growing need for advanced network intrusion classification and introduces the deep reinforcement learning approach as a solution to address the shortcomings of traditional methods. In chapter II, the background provides foundational knowledge on deep reinforcement learning set up and data in the context of network security. In chapter III, the related work reviews existing literature on intrusion classification, identifying gaps in prior research and highlighting the need for the proposed DRL-based model. In chapter IV, the methodology outlines the design of the DRL-based NIC, detailing the architecture, datasets, and techniques used for feature selection and model training. In chapter V, result analysis presents experimental findings, comparing the proposed model's performance with existing methods, showcasing improvements in detecting known and novel attacks. In chapter VI, the testing section evaluates the model's adaptive capabilities using the NF-BoT-IoT dataset, demonstrating its effectiveness in detecting previously unseen threats. In chapter VII, limitations are acknowledged, such as the complexity of implementation and the computational demands of training deep reinforcement learning models. The conclusion summarizes the contributions, highlighting the effectiveness of the DRL-based NIC and suggesting future research directions for further improvement.

Chapter 2

Background

2.1 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning comprises a sophisticated approach, combining two of the most powerful fields in artificial intelligence: reinforcement learning and deep learning. Reinforcement learning is a paradigm of machine learning wherein an agent learns to make decisions through interaction with an environment. The agent seeks to maximize some notion of cumulative reward through trial and error, whereby its behavior is modified based on feedback received from the environment. This includes taking actions in different states of the environment and getting the outcomes from these, which become helpful for later decision making. With time, the agent develops its own strategy, called policy, to act more successfully with regard to its goals [9]. Deep learning, on the other hand, is a subset of machine learning that makes use of multi-layer neural networks, hence giving the system capabilities to model and process highly complex data structures such as images, text, and sequences. Employing deep learning, DRL allows agents to handle environments where observations are high-dimensional input data, including long sequences in time-series data. In DRL, the neural networks are used to approximate value functions or policies that enable the agent to generalize over very large state and action spaces, and as such, enable the solving of hard problems that are otherwise intractable by traditional RL methods. Deep reinforcement learning integrates the decision-making framework of reinforcement learning with the rich representation capabilities of deep learning and, therefore, is one of the main techniques used for solving complex tasks. In the process of DRL, the agent receives feedback as rewards or penalties based on the actions that it performs, thus enabling it to improve its performance over time. Before diving into DRL, it is crucial to understand the basic components of Reinforcement Learning :

- Agent: The learner or decision-maker that interacts with an environment.
- Environment: The external system the agent interacts with, which provides feedback.
- State: The current situation of the agent in the environment.

- Action: The decisions the agent can make.
- Reward: A scalar feedback signal received after taking an action in a particular state. The goal of the agent is to maximize the cumulative reward over time (also known as the return).
- Policy: A strategy that the agent follows to choose actions based on its current state.
- Value Function: A function that estimates how good a particular state or action is in terms of long-term rewards.
- Q-value (Action-Value): The expected return of taking an action in a given state and following a policy thereafter.

The agent interacts with the environment in discrete time steps: at each step, it observes the state, selects an action according to its policy, receives a reward, and transitions to a new state. This continuous feedback loop helps the agent improve its policy over time.

2.1.1 Network Intrusion Classification

DRL can significantly enhance Network Intrusion classification to independently adapt with changing cyber threats. Traditional NIC approaches are typically based on predefined signatures or rule-based systems, where specific patterns or behaviors associated with previously known threats are encoded into the system. While effective against known attack types, these traditional methods could prove limiting towards new and more sophisticated attack vectors; this is particularly problematic when the threats have not been seen before, often termed as zero-day attacks. By nature, signature-based systems are static and often reactive, thus allowing systems to remain mostly vulnerable against evolving and increasingly complex attacks that do not match established patterns. The solution provided by DRL is dynamic and adaptive. Unlike the static NIC models, DRL agents learn over time by interacting with the environment-in this case, network traffic-by a process of trial and error. The agents observe and monitor network traffic, continuously adjusting their strategies and improving the capability of anomaly detection as they are exposed to new types of data. Instead of using only fixed signatures or rules, DRL-based NIC systems can learn autonomously to identify abnormal traffic patterns indicative of malicious activity, even when those patterns differ significantly from any previously known attacks.

With regard to NIC, one of the main strengths of DRL is its ability to deal with high-dimensional complex data, such as network traffic logs. Thus, by training on large and diverse datasets of both normal and malicious network traffic, DRL models will be able to learn to differentiate between benign behavior and various forms of attacks. The neural networks applied in DRL will extract detailed patterns from data, thus enabling the system to build very rich and layered representations of network traffic. This will provide a better chance for DRL-based NIC to detect subtle variations that indicate a cyber-attack,

including previously unseen attack patterns that might be overlooked by traditional IDS. It can be very adaptive, allowing DRL-enhanced NICs to improve their detection over time. Finally, as the DRL agent continuously interacts with a live environment, it learns through continuous improvement of its policy by updating new and emerging threats. This practice becomes effective in addressing the challenge of zero-shot attacks, where new attack vectors are introduced to counter defenses that have not been tried or documented. The DRL-based NIC shows a better and proactive approach toward cybersecurity by learning and adapting in real time, developing its resilience against known and emerging threats.

2.1.2 Adaptive Learning by DRL

Deep reinforcement learning can offer a really efficient adaptive response strategy in case intrusion is detected. Such limitations of traditional security mechanisms are the fact that they apply a kind of pre-defined response mechanism, which may seem to be insufficient or even inflexible for a wide range of potential threats a network could face. In contrast, DRL enables the system to make dynamic decisions on the most adequate response based on given characteristics and severity of the threat and the current state of the network. For example, when a security breach is detected or malicious activity is predicted, the DRL agent can be trained to evaluate such a situation and decide the best action.

In minor cases, it may learn one or a few kinds of actions according to the type of intrusion. The agent can only block the incoming traffic from some IP addresses associated with the suspicious traffic and avoid further malicious access. But this decision is not made arbitrarily; rather, it's influenced by prior experiences and training of the agent on similar network conditions and threats. In critical situations where an attack may have far-reaching implications, the DRL agent can determine necessary interventions. Such interventions can be in terms of routing network traffic over alternative routes to ensure uninterrupted provision of essential services when parts of the network are being attacked.

In addition, the agent can take rate-limiting measures by reducing the bandwidth or limiting the number of requests coming from suspicious sources. This mechanism prevents Distributed Denial of Service (DDoS) and other threats that occur in high volume from causing much damage-throttling such malicious traffic while allowing legitimate users access to the network resources. In addition to such an immediate response, DRL can contribute even in the higher-order task of resource deployment optimization across a network during an attack. For instance, while an attack is in progress, the network must be efficient and, if needed, declare its defense. A DRL agent may be designed to understand which parts of the network are most critical during the attack, and, accordingly, focus the allocation of additional security. For example, the agent will be able to identify which server or device is at a higher risk to be isolated from the other network so it will not propagate malware and leak information.

Should the attack be highly focused, the DRL system can go all the way to determine which network parts should be shut down or temporarily isolated in order to reduce the

scope for widespread damage. It learns to take such decisions through balancing the trade-offs between preserving operational functionality and the prevention of further attack escalation. The power of adaptability ensures an effective response and one optimally designed to minimize its impact on business critical operations. DRL can be used to continually modify the response strategy over time. With experience gained by the agent, responding against other types of intrusion, the agent further refines its decision process and thus becomes more efficient in choosing responses that minimize further damage, make effective resource utilization, and preserve the functionality of key network components. This makes DRL a potent method for developing smart, independent cybersecurity systems capable of adapting to an ever-changing threat landscape.

2.1.3 Vulnerability Management

DRL can be extremely useful in such environments to help prioritize remediation efforts. Networks-especially large and complex ones-expose themselves to a wide range of security vulnerabilities, from minor issues to critical weaknesses that could be potentially exploited by attackers. Not all of them can be fixed at the same time due to time and resource constraints. Thus, organizations have to determine which vulnerabilities present the greatest risk to a system and therefore should be fixed first. DRL provides an advanced, data-driven mechanism to help make these prioritization decisions more effective.

Treating the network environment as a dynamic system, DRL enables the development of an intelligent agent capable of modeling the interrelations between various types of vulnerabilities, the possibility of their exploitation, and the overall impact they can potentially have on the network. It ultimately learns to look at each identified vulnerability from the perspective of the entire network, while understanding how likely a particular vulnerability is to be successfully exploited and what it would actually mean if it were attacked. With this holistic network-level view, the DRL agent evaluates the vulnerabilities based on various factors like their actual severity, the probability of successful exploitation, and the criticality of network assets that they affect.

As the DRL agent interacts with the network environment, it can simulate different attack scenarios, learning from these interactions to predict which vulnerabilities would have the most detrimental impact if left unaddressed. For instance, a vulnerability in a highly critical server that stores sensitive data might pose a far greater risk than a similar vulnerability in a less critical system. Through this process of trial and error, the DRL agent can refine its understanding of how different vulnerabilities affect the overall security posture of the network.

It can then recommend a prioritized list of vulnerabilities to remediate once an agent has developed a clear understanding of the potential risks. A prioritized list based on all such risk assessments wherein vulnerabilities that would pose more risk either because of their higher likelihood of exploitation or because of the high impact due to such exploits are accorded the highest priority. By focusing on the high-risk vulnerabilities first, orga-

nizations can therefore manage their resources more effectively, putting the most critical security concerns into focus while minimizing the possibilities of potential attacks. DRL agent is not static in its assessment. As new vulnerabilities emerge or as the network configuration changes, the agent can continuously update its prioritization model. This agility makes the organization responsive to evolving threats and changing conditions, focusing its remediation efforts where they are most needed. Thus, DRL enables security teams to outsmart attackers by prioritizing and mitigating those vulnerabilities that could most likely be used in an attack. In environments containing thousands of vulnerabilities, it becomes an impossible task for an analyst to go through each and prioritize them manually. In this context, DRL automates a lot of the decision-making, freeing security professionals to implement the fixes, rather than having key time wasted on analyzing risk. This would make the whole vulnerability management more proactive and effective to handle, where the major risks are dealt with quickly, therefore reducing the overall exposure of the network to potential attacks.

2.2 DRL's Adaptive Modeling in NIC

By adaptive modeling in DRL, it is meant that a model has the dynamic capability to change its behavior and generalize to new, unseen data patterns during testing or deployment. This is one of the key strong points of DRL compared to more traditional machine learning models, which usually have to rely heavily on the training data and might underperform when faced with either novel or evolving settings. In DRL, the agent continuously learns by interacting with its environment, thereby gaining experience, and receives feedback in terms of rewards or penalties. By doing so, the agent dynamically updates its policy regarding decision making to adapt to any change in the surroundings, especially in those conditions where it is quite difficult to predict the correctness of the decision beforehand or when the environment undergoes rapid changes. This is a very valuable feature of adaptive modeling, especially in NIC, given the dynamic nature of the threat landscape. Cyberattacks are becoming increasingly complex and varied; new attack vectors can appear at any moment. Static or rigid models that depend on predefined signatures or patterns from historical data can be quite ill-equipped to handle such new kinds of threats. Such models may miss zero-shot attacks or other sophisticated attacks with patterns different from what they have been trained for. This is where DRL becomes important with its adaptive modeling capability.

In this case, the environment of a NIC is network traffic, which keeps changing over time with new protocols, user behaviours and methods of attack. During training, the DRL model could be exposed to various data patterns, whether normal or malicious in nature, so that the agent learns to differentiate between benign and harmful activity. However, the real strength of DRL comes during the testing or deployment phase when new data patterns are observed that the model has never seen before. Due to its adaptive learning nature, the DRL model generalizes its understanding from previous experiences

and applies them to these new situations, where informed decisions must be made on whether the network activity observed is suspicious or benign. This flexibility results from the capability of an agent to learn continuously through interaction with its environment. A DRL-based NIC, in comparison with other conventional models that tend to go obsolete immediately upon emergence of new attack vectors, can improve its decision-making policy progressively by utilizing new incoming data and feedback. It can even, to some degree, adapt its behavior when faced with novel attack patterns that have not been explicitly present in the training data. A good example is when the agent may observe some kind of network traffic that was unseen before but shares certain characteristics in ways similar to known malicious behaviors; thus, it adjusts its action in either flagging the traffic as suspicious or taking precautionary steps like blocking or limiting the traffic. In this case, the DRL-based NIC enables an adaptive learning process to outsmart such evolving threats. Due to the increased complexity of cyberattacks, or the development of new techniques by attackers to circumvent traditional security controls, DRL models can handle these shifting dynamics more effectively. Because of the capability of continuous improvement in its decision-making processes in real-time, the area of DRL becomes powerful in defense mechanisms on a wide scale for zero-shot exploits, among other unforeseen attack vectors.

Consider a scenario shown in Fig.2.1 where network flow data is divided into two categories: benign data (labeled as 0) and various forms of malicious attacks (labeled as 1). These attacks may include:

- Backdoor
- Denial of Service (DoS)
- Distributed Denial of Service (DDoS)
- Injection attacks

Suppose the training dataset contains benign data (labeled 0) and two attack types: Backdoor and DoS (labeled as 1). The testing dataset, however, includes all attack types, including previously unseen attack categories like DDoS and Injection. In this case, the DRL model is trained to distinguish between benign and malicious behavior based on the data available in the training set, but it has not been exposed to all possible attack types. Despite this, the DRL model can still perform well on the test set that includes new types of attacks, such as DDoS and Injection. This is where adaptive modeling comes into play.

2.2.1 DRL's Strength in Handling Unseen Attacks

Generalization through Policy Learning: DRL models do not merely memorize patterns seen during training; they learn a policy, a set of actions or decisions, based on the broader characteristics of network traffic behavior. These policies are trained to maximize rewards (successful detection of intrusions) by adapting to various network conditions, making the model capable of generalizing to unseen attack types. Even though DDoS

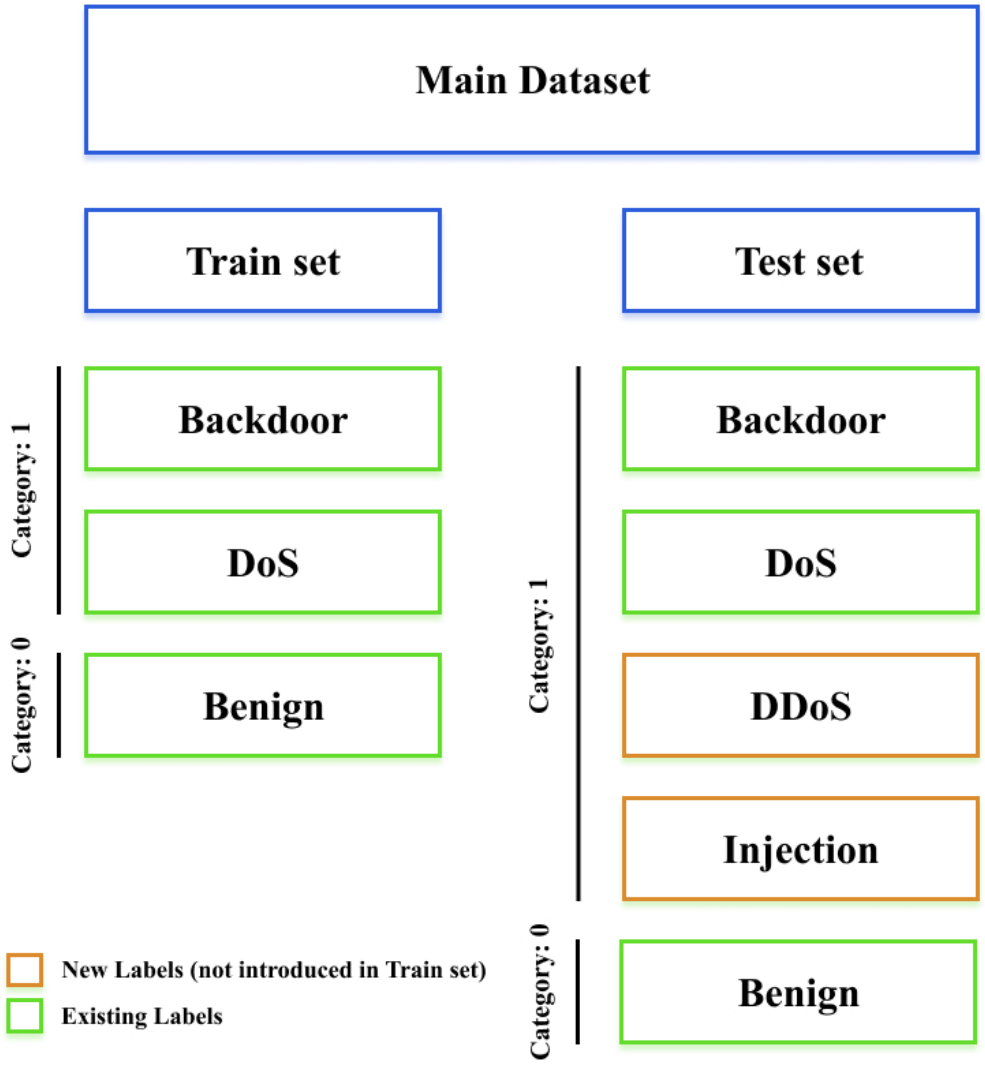


Figure 2.1: Adaptive Modeling Data distribution Strategy.

and Injection attacks were not part of the training data, the model can recognize patterns similar to previously encountered attacks (Backdoor, denial-of-service (DoS)), utitilizing its policy to identify these new threats as anomalies or malicious.

Exploration and Exploitation: In DRL, the exploration-exploitation trade-off allows the model to explore different actions and outcomes in its training phase. This enables the model to adapt to new attack vectors in the test set, as it can identify similarities in network flow anomalies, even if the precise attack type (e.g., DDoS or Injection) was not in the training data. Through exploration, the DRL model learns features that are common across various attack types and exploits this knowledge to detect unseen threats.

Feature Extraction: Deep learning models, including DRL, are highly effective at extracting important features from raw network flow data. DRL can learn abstract representations of network behaviors, enabling it to differentiate between benign and malicious

activities even when new types of attacks arise. For instance, DDoS and Injection attacks might have different signatures than Backdoor or DoS, but they may share certain characteristics such as abnormal traffic spikes, unusual request patterns, or irregular data injections, all of which the model can pick up during testing.

Adaptation via Continuous Learning: DRL models are inherently capable of continuous learning through interaction with their environment. Although the training phase is fixed, the DRL model's architecture allows it to adapt to new situations by leveraging its understanding of previously encountered malicious behavior. This adaptive quality is crucial for dynamic environments like networks, where new types of attacks are constantly emerging.

We have conducted Adaptive modelling testing for our proposed model discussed in chapter VI.

2.3 Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks are a type of architecture for recurrent neural networks (RNNs) that are well suited to modeling sequential data. Traditional RNNs suffer from the problem of learning long-range dependencies because of phenomena like vanishing gradients. To solve this constraint, LSTMs have introduced a new memory cell capable of storing information for an extended period of time, thus providing an excellent way to handle intricate temporal relationships residing in data [10]. LSTMs are particularly good at time-series analysis because learning sequences of data where order matters a lot suits them. This would make LSTMs very good for capturing temporal dependencies in network traffic data that might go unnoticed by traditional models. Network traffic data is sequential, consisting of packets and events that occur over time. LSTMs can analyze these sequences to identify patterns and anomalies, which may be useful for intrusion detection. Network behavior can change over time in a sequential manner, showing trends and changes in, for example, user activity or attack strategies. LSTMs are really good at modeling these long-term dependencies, enabling the detection of even very minor shifts in traffic that could serve as possible indicators of something suspicious.

2.4 Deep Neural Networks (DNN)

Deep Neural Networks (DNNs) are an artificial neural network with more layers of nodes (neurons) sandwiched between the input and output layers. With interconnected neurons in every layer, each layer transforms the input data using weighted connections and activation functions. Though DNNs are strong in many machine-learning tasks, their architecture is essentially designed for static data inputs [11]. With network traffic, the data are inherently sequential and temporal; thus, the order in which the packets arrive and the time offset between them could be of vital importance to detect patterns—something that is usually seen when trying to detect intrusions. DNNs excel at automatically extracting

complex features from raw data. In network traffic, features like packet size variations, frequency of connections, and the timing of requests can indicate normal behavior or potential attacks. DNNs can capture these intricate relationships without the need for extensive manual feature engineering. DNNs can recognize patterns in the data that may not be immediately apparent. In the context of intrusion detection, this means they can learn to identify the characteristics of both normal traffic and various attack patterns (e.g., DDoS, malware, unauthorized access) from historical data. Although DNNs are typically trained on large batches of data, they can be adapted for real-time analysis through techniques like mini-batch training or online learning. This adaptability is crucial in network intrusion detection, where the system needs to identify threats as they occur, responding dynamically to changing traffic patterns.

While DNNs provide robust capabilities, they can be enhanced when combined with other architectures, such as LSTM networks. LSTMs are particularly well-suited for time-series data due to their ability to retain information over longer periods, making them ideal for analyzing sequences of network packets. A hybrid approach, combining DNNs for feature extraction with LSTMs for temporal analysis, can significantly improve the performance of intrusion classification.

2.5 Analyzing Network Traffic Patterns

We have used NF-Bot-Iot as primary dataset. Let us understand the data and features before diving into intrusion detection. NF-Bot-IoT [12] dataset represents network traffic information, capturing various aspects of packets exchanged between devices in a network. The dataset captures both normal (benign) traffic and malicious activity (thefts). By analyzing these patterns, it becomes possible to identify anomalies and potential threats. The presence of various transport layer protocols (TCP, UDP) indicates the diversity of applications in use. Notably, common ports such as 80 (HTTP) and 53 (DNS) are frequently encountered, which reflects typical web and network behavior. Fields such as IN_BYTES and OUT_BYTES can be used to analyze the amount of data transmitted during each session. This is critical for assessing the impact of specific traffic flows on network performance. TCP_FLAGS and FLOW_DURATION_MILLISECONDS provide insights into the connection dynamics, helping to distinguish between normal handshakes and potentially malicious patterns like SYN floods.

2.5.1 Detailed feature Descriptions

- IPV4_SRC_ADDR: This column represents the IPv4 address of the source device that initiates the data flow. IPv4 addresses are numerical labels assigned to each device connected to a computer network that uses the Internet Protocol for communication.
- L4_SRC_PORT: The source port number used by the application layer on the originating device. It identifies the specific service or process generating the traffic. Each

service on a device is assigned a unique port number. For instance, web servers typically use port 80 for HTTP. Monitoring source ports can help in identifying unusual traffic patterns, such as services that are not typically accessed or unexpected connections.

- IPV4_DST_ADDR: This is the destination IPv4 address, indicating the target device that is receiving the data. Similar to the source address, the destination IP is crucial for mapping traffic flow within the network. It helps in identifying which devices are being accessed or attacked and can assist in pinpointing compromised devices.

- L4_DST_PORT: The destination port number at the transport layer, specifying the service or application on the target device that is intended to receive the data. Analyzing destination ports helps in understanding what services are being accessed. For example, if multiple flows target an unusual port, this might indicate probing for vulnerabilities or an ongoing attack.

- PROTOCOL: This represents the application layer protocol being utilized for the traffic flow, also denoted numerically. The application layer protocol provides insights into the type of service being accessed (e.g., HTTP, FTP). This helps in categorizing the traffic and understanding user behavior, as well as identifying any unusual or unauthorized applications in use.

- IN_BYTES: This column records the total number of bytes received by the source device during the data flow. Monitoring inbound traffic volume is critical for assessing network load and can highlight potential issues such as denial-of-service attacks, where a device may receive an abnormally high amount of data.

- OUT_BYTES: The total number of bytes sent out from the source device during the data flow. Similar to IN_BYTES, analyzing outbound traffic helps to monitor data ex-filtration attempts, where sensitive data may be sent outside the organization.

- IN_PKTS: The count of packets received by the source device during the flow. Packet counts can provide insight into the nature of the traffic. For instance, a high number of packets with low byte size might indicate a scanning or probing activity.

- OUT_PKTS: The total number of packets sent out from the source device. This metric is crucial for understanding how much data is being communicated and helps in identifying unusual patterns that could indicate malicious behavior.

- TCP_FLAGS: This field represents various flags set in the TCP header, such as SYN, ACK, FIN, and RST. These flags control the state of the TCP connection. Analyzing TCP flags is important for diagnosing connection states. For example, a high number of SYN packets without corresponding ACKs could suggest a SYN flood attack, indicating an attempt to overwhelm a service.

- FLOW_DURATION_MILLISECONDS: The duration of the data flow in milliseconds, indicating how long the flow lasted from start to finish. Flow duration helps in assessing the nature of the communication. Very short flows might indicate scanning, while longer flows could represent normal sessions. This can also help in identifying potential issues like timeouts or abrupt connection drops.
- Label: This categorical label provides an indication of whether the flow is considered benign or malicious (e.g., 'Theft').
- Attack: A binary indicator (0 or 1) that signifies whether the flow is associated with malicious activity (1) or is benign (0).

2.6 Related Work

In this section, the related work reviews existing literature on intrusion detection systems, identifying gaps in prior research and highlighting the need for the proposed DRL-based model. Author selected papers from different network environment based NIC work as this study is focused on developing network intrusion classifier which can adapt on variety of network flow data by the help of DRL framework.

Table 2.1: Comparison of Network Intrusion Classification Based on Key Features.

Paper Ref.	**Class Imbalance Handling**	**Adaptive Modeling**	**Unseen Threat Detection**	**Handling Evolving Attacks**
[13]	No	No	No	No
[14]	No	No	No	No
[15]	No	No	No	No
[16]	No	Yes	Yes	Yes
[17]	Yes	Yes	Yes	Yes
[18]	No	No	No	Yes
[19]	No	No	No	Yes
[20]	Yes	Yes	Yes	Yes
[21]	No	Yes	No	Yes
[22]	No	Yes	No	Yes
[23]	No	Yes	Yes	Yes
[24]	No	Yes	Yes	Yes
[25]	No	Yes	Yes	Yes
[26]	No	Yes	No	Yes
[27]	No	Yes	No	Yes
Proposed System	**Yes**	**Yes**	**Yes**	**Yes**

Recent progress in AI, more so DRL, has proved to be quite capable of improving the efficiency of NID. H. Benaddi et al. [13] have demonstrated the capability of RL in finding abnormal network access and attacks, a crucial task toward robust network se-

curity. Additionally, J. Lansky et al. and Z. Wang et al. [14, 15] has underscored the efficacy of DRL in addressing these complex security issues. Current NID face significant challenges in detecting sophisticated and evolving cyber threats. Traditional intrusion detection methods often struggle with the complexity and volume of network data, leading to issues such as high false negative rates and the inability to detect novel attack patterns. Machine learning-based approaches, including network packet classification, have improved the performance and security of networks, but there remain gaps in their ability to handle large-scale, complex data and sophisticated attacks [28]. DRL based approach showed promising result, M. He et al. [16] proposed a transferable and adaptable network intrusion detection system (TA-NID) based on deep reinforcement learning. Their system improves robustness and adaptability by using small-scale datasets to generate diverse interactions, prioritizing outliers without requiring full dataset classification, and ensuring model transferability across different datasets. Experimental results showed high accuracy and effective prioritization of outliers. S. Dong et al. [17] proposed network abnormal traffic detection model Based on Semi-Supervised Deep Reinforcement Learning. The abnormal traffic detection can be further optimized with the utilization of the SSDDQN-based optimization approach. The core of this approach is how DDQN is applied to enhance the efficiency and accuracy of learning. An autoencoder was utilized to conduct feature reconstruction, thereby allowing the current network to conduct dimensionality reduction with the preservation of the key features of traffic. This is crucial for the processing of high-dimensional data where only the most important features remain for further analysis. Following feature reconstruction, accuracy in detection is enhanced through the classification by a deep neural network classifier. This classifier is enhanced as a result of the capability of the autoencoder on dimensionality reduction, which makes it concentrate more on the most important aspects of the traffic data. Besides, unsupervised learning is also included in the target network by incorporating K-Means clustering in order to group similar data into meaningful clusters. The DNN uses these clusters for making the prediction, thereby combining strengths from both unsupervised and supervised learning to further enhance the detection capability of abnormal traffic.

T. Kim and W. Pak [18] suggested approach involves generating a fresh training dataset for Generative Adversarial Network by leveraging misclassified data sourced from a bespoke training dataset via an LSTM-DNN model trained on the original dataset. There is hardware dependency and accuracy need to be further improved, compared to the conventional approaches. Machine learning based for network intrusion detection proposed by R. R. d. Santos et al. [19], achieving high accuracy but struggling with evolving traffic patterns over time. Commonly, these models require frequent updates, posing practical challenges. A novel model, using reinforcement learning for long-term reliability and classification accuracy, addresses these issues. This approach combines transfer learning and a sliding window mechanism to minimize computational demands and manual interventions. Experiments with an 8TB, four-year dataset reveal that traditional models less effective with dynamic traffic, whereas the proposed RL model maintains similar accuracy without

frequent updates. Periodic updates further reduce false negatives, using significantly fewer resources and only seven days of training data. X. Ma and W. Shi [20] introduced a novel framework for anomaly detection in IDS, combining RL with class-imbalance techniques to improve detection accuracy. The methodology included an adapted Synthetic Minority Over-sampling Technique (SMOTE) to address dataset imbalance, enhancing the RL agent's performance. Experiments using the NSL-KDD dataset demonstrate the model's efficacy. Comparative evaluations highlight that the proposed AESMOTE model outperforms AE-RL, achieving an accuracy above 0.82 and an F1 score above 0.824. However, limitations include reliance on the NSL-KDD dataset, which may not represent all real-world scenarios. Multi-dataset validation can help to prove proposed solution's robustness in terms of real-world scenarios.

MalBoT-DRL, a novel deep reinforcement learning-based botnet detector proposed by M. Al-Fawa'reh et al. [21], it allows dynamic adaptation to changing malware patterns. Validated on MedBIoT and N-BaIoT datasets, MalBoT-DRL achieved impressive detection rates of 99.80% and 99.40% in early and late detection phases, respectively. However, limitations include challenges in detecting stealthy malware like Bashlite, vulnerability to adversarial attacks, and increased latency with larger datasets. Despite these, the model maintained consistent training efficiency. Enhancing exploration-exploitation policies and employing network distillation can mitigate these challenges, ensuring robustness and reduced computational demands for real-world IoT environments. C. Picard and S. Pierre [22] proposed RLAuth, a risk-based authentication system using DRL to classify authentication contexts as anomalies or regularities. The methodology includes an Anomaly Detector and a Risk Engine to compute authentication risk based on context, previous authentication data, and application sensitivity. An Authentication Manager dynamically selects appropriate authentication methods. However, RLAuth is vulnerable in familiar contexts, such as attacks by coworkers or family members, and arbitrarily chosen confidence levels and application sensitivities could be optimized using mathematical or machine learning models. Additionally, storing context information on device memory leads to high consumption over time, suggesting a need for efficient modeling techniques. Despite these limitations, RLAuth enhances mobile application security by balancing usability and privacy, dynamically adapting to new environments. The Anomaly Detector achieved a G-Mean of 92.62% on the testing set, demonstrating the effectiveness of DRL in user authentication.

The study by [23, 29] proposed a reinforcement learning-based collaborative DDoS detection method, using edge computing for early detection and efficient resource utilization. The methodology involved deploying lightweight unsupervised classifiers in IoT edge gateways to analyze network traffic features in real time. The soft actor-critic (SAC) reinforcement learning model dynamically adjusts classifier parameters, ensuring robust detection across various IoT devices. A collaborative aggregation module enhances detection by sharing states and experiences. Experiments on public datasets and a real-world dataset show excellent detection performance, accurately identifying stealthy IoT-based

DDoS attacks. Limitations include potential adaptation delays in highly dynamic environments. Intrusion detectors are increasingly vulnerable to adversarial attacks, with current countermeasures often lower performance and being algorithm specific. G. Apruzzese et al [24] proposed a novel framework using DRL to protect botnet detectors from adversarial attacks. The methodology involves generating realistic adversarial samples that evade detection and using these samples to create an augmented training set, producing more resilient detectors. Extensive experiments on several machine learning algorithms and publicly available datasets validate the effectiveness of the proposed method. The results show significant improvements over state-of-the-art methods while being robust to unforeseen attacks and preserving performance in situations without adversarial manipulations. Key limitations include potential computational complexities as well as the requirement for heterogeneous training datasets.

The Industrial Internet of Things presents several cybersecurity risks, calling for advanced intrusion detection systems. S. Yu et al. [25] proposed a DC-IDS solution for IIoT using DRL, modelling the open-set recognition problem as a discrete-time Markov decision process. A DQN with a conditional variational autoencoder handles the known traffic classification and unknown attack recognition. Known traffic classification is handled by DQN while unknown attacks are recognized by reconstruction error. Investigations utilizing the TON-IoT dataset illustrate the efficacy of the DC-IDS model, which attains enhanced performance in identifying unfamiliar attacks while maintaining model stability relative to previous approaches. Potential constraints may encompass scalability issues associated with extensive IIoT implementations as well as the necessity for considerations regarding real-time application. Progressions in IoT and cloud computing introduce difficulties for intrusion detection within enterprise networks.

S. Vadigi et al. [26] proposed a Federated DRL-based IDS, deploying multiple agents with DQN logic across the network. Their system preserved data privacy by keeping data at each agent node local while utilizing an attention-weighted model aggregation process to share learning. An attention mechanism dynamically determines each agent's contribution to model aggregation. Tested on the ISOT-CID and NSL-KDD datasets, their system demonstrates robust performance with high accuracy, precision, and a low false-positive rate. Limitations include potential computational complexity and the need for effective handling of diverse network environments.

The work by L. Chavali et el. [27] introduced TD3-AP and SAC-AP, two DRL-based off-policy actor-critic methods for alert prioritization, addressing the instability and limited exploration of existing DDPG-based methods. Using twin delayed deep deterministic policy gradient (TD3) and soft actor-critic (SAC), the interaction between adversary and defender is modeled as a zero-sum game with a double oracle framework for mixed strategy Nash equilibrium. Extensive experiments on MQTT-IoT-IDS2020, DARPA 2000 LLDOS 1.0, and CSE-CIC-IDS2018 datasets show TD3-AP and SAC-AP reduce defender's loss by 50% and 14.28%, respectively, outperforming DDPG and traditional methods. The results' interpretability is enhanced using SHapley Additive explanations (SHAP). Limita-

tions include sensitivity to hyperparameters and the need for significant training time and computational resources. C. Rookard and A. Khojandi [30] introduced RRIoT, utilizing a Deep Deterministic Policy Gradient (DDPG) reinforcement learning algorithm combined with an LSTM layer in an adversarial setting to enhance attack detection and identification. Compared to existing ML and RL algorithms like DQN, DDQN, and DDPG, RRIoT demonstrates superior performance. Evaluations using three IoT telemetry datasets show that RRIoT outperforms state-of-the-art ML algorithms and matches or exceeds novel RL algorithms. Additionally, Shapley Additive Global Importance (SAGE) is used to identify key contributing features, with feature importance confirmed through an ablation study. Limitations may include computational complexity and resource requirements for training the model. Lastly, the use of DRL in NID is particularly advantageous due to its ability to process and learn from vast amounts of data, identify complex patterns, and adapt to new and unseen threats. This makes DRL-based approaches well-suited to the demands of modern network security, where traditional methods often fail. As cyber threats continue to evolve, the integration of DRL into NID represents a significant advancement, offering enhanced detection capabilities and improved resilience against sophisticated attacks. Notably, intrusion detection using DRL techniques is still in development but shows great promise [31, 32]. Researchers often aim for higher classification accuracy by modeling the intrusion detection domain as an DRL-based environment. Therefore, it is evident that DRL-based approaches are not only fitting but also necessary for the future of network intrusion detection.

A growing body of research highlights the significant advancements in applying GANs to intrusion detection systems. The work in [33] introduced a projection-based adversarial attack generation for IDS using a traffic space GAN to model the distribution of malicious and benign traffic. Similarly, [34] applied GANs to generate synthetic data, followed by classification using deep neural networks, convolutional neural networks (CNN), and long short-term memory, demonstrating that the GAN+DNN combination provided superior performance on the UNSW-NB15 dataset. Studies like [35] have addressed challenges in GAN training, such as instability and mode collapse, by proposing ensemble-based multi-layer GANs (EMP-GANs) that enhance training stability and synthetic data diversity. Meanwhile, traditional AI methods like decision trees [36] and support vector machines [37] have been widely explored in early network intrusion detection efforts. A comparative study in [38] evaluated Naïve Bayes, SVM, and K-Nearest Neighbors on NSL-KDD and UNSW-NB15 datasets, revealing varied results in intrusion detection accuracy across algorithms.

Deep learning approaches continue to gain traction in this domain, as demonstrated in [39], where DNNs showed their potential in efficiently classifying network threats. Zhong et al. [40] introduced a system integrating big data and tree structures with deep learning, yielding better performance compared to traditional methods. authors in [41] developed a method using GAN to produce adversarial attack data and proposed a robust IDS model (RAIDS) to counter these attacks, illustrating the evolving nature of threats and the

need for adaptive defense mechanisms. In another study, [42] proposed a Convolutional Deep Belief Networks (CDBN) model for detecting intrusions in wireless networks, which, despite its accuracy, is resource-intensive due to the complexity of its architecture. Recent hybrid models, like the CNN-BiLSTM architecture discussed in [43, 44], further illustrate the potential of combining different deep learning techniques to handle both binary and multiclass intrusion detection, achieving an average accuracy of 84.42% on the NF-UNSW-NB15 dataset. Haghighat et al. [45] proposed an intrusion detection system based on deep learning and voting mechanisms that aggregate the best models for more accurate and robust results, effectively leveraging ensemble learning to counter diverse intrusion tactics.

Anomaly-based intrusion detection was contributed by [46], which proposed the Locality-Sensitive Hashing-based Isolation Forest model. With adaptive learning techniques, this model evolves continuously against evolving network conditions and is found to be very effective in highly variable environments. The system also applies window sliding and dynamic model updates of the features that may help in sustaining high accuracy over time. It is flexible enough to detect new and known threats efficiently while the nature of network traffic is continuously changing. This approach was tested with the widely used KDDCUP99 dataset and showed significant improvements in detecting anomaly activities compared to traditional approaches. It highlights the potential that adaptive learning bears with respect to the improvement of evolved cyber-attack detection. The proposed HC-DTTWSVM methodology for network intrusion detection by authors from [47] requires data collection and data preprocessing of network traffic from benchmark datasets like the NSL-KDD and UNSW-NB15, ensuring the quality of data by cleaning and normalization with feature extraction. Further, a bottom-up hierarchical clustering algorithm arranges this into a decision tree that optimally separates different intrusion categories. The TWSVMs are then integrated into this tree for fast classification of network traffic. The model will be trained using cross-validation techniques and tested against various state-of-the-art techniques on relevant performance metrics accuracy and F1-score. However, the approach suffers from some setbacks, including dependence on the quality of data, scalability issues when the dataset gets bigger, challenges in feature selection, and reduced generalization to new types of attacks not present in the training data. Moreover, to achieve the best result, hyperparameter tuning is required, and embedding TWSVM complicates the interpretability of the model, which may challenge stakeholders who require transparency in decision-making processes.

Similarly, the work in [48] demonstrates the application of Generative Adversarial Networks to improve the performance of intrusion detection systems in Internet of Things networks. Given the rapid expansion of IoT devices and the increasing diversity of potential security threats they face, traditional intrusion detection models often struggle with imbalanced datasets, where certain types of attacks are underrepresented. To address this, the authors employed GANs to generate synthetic data for these underrepresented attack classes, effectively balancing the dataset. This augmentation process resulted in a

significant performance boost, improving the accuracy of the IDS from 84% to 91% when tested on the UNSW-NB15 dataset. The use of GANs in this context not only enhances the model's ability to generalize across different types of intrusions but also underscores the broader trend toward leveraging synthetic data generation in network security.

These advances emphasize the growing importance of deep learning techniques, particularly GANs, in addressing the dynamic and ever-evolving landscape of network threats. The ability to generate realistic synthetic data and adapt to changes in network traffic patterns represents a crucial advantage in modern IDS development. As cyber-attacks become more sophisticated, techniques like those presented in [46] and [48] provide essential tools for detecting anomalies and securing networks in a wide variety of contexts, from traditional networks to the more complex and heterogeneous environments of IoT systems.

The table 2.1 presents a comparative analysis of various studies focused on Network Intrusion classification based on key features such as class imbalance handling, adaptive modeling, unseen threat detection, and the ability to manage evolving attacks. Each row corresponds to a different research paper, with the columns indicating whether the respective study effectively addresses these critical challenges. For instance, the works by H. Benaddi et al. [13], J. Lansky et al. [14], and Z. Wang et al. [15] do not utilize any of the specified capabilities, suggesting that these approaches may be less effective in practical applications where cyber threats are continually evolving. M. He et al. [16] present an approach that could handle both the above challenges and include some form of adaptive modeling and unseen threat detection. It also fails to address the issues of class imbalance and management of evolving attacks. Similarly, the approach proposed in works like those by T. Kim and W. Pak [18], S. Vadigi et al. [26] suffers from either class imbalance or unseen threat detection challenges. It is observed that the work of L. Chavali et al. [27], while it may be efficient in adaptive modeling, does not have the capability of unseen threat detection, which again shows the further gaps in the existing methodologies. Of particular note, the final row represents the proposed system, which effectively manages all four features in an indication of comprehensiveness against the shortcomings identified above. It's indicative that the proposed system may result in key advances in intrusion detection, considering that solutions are desperately needed to be adaptable under shifting network conditions, handle class imbalances, and find unseen and evolving threats.

2.6.1 Dealing with Limitations

This work addresses key limitations found in prior studies by lutilizing a novel deep reinforcement learning-based approach, integrating a hybrid model combining DNN and LSTM networks, along with advanced data balancing techniques. H. Benaddi et al. [13], J. Lansky et al. [14], and Z. Wang et al. [15] focus on traditional machine learning models that often struggle to adapt to evolving cyber threats. They rely on static datasets and pre-defined attack patterns, limiting their effectiveness in detecting novel attacks and

unseen threats. This work's DRL-based model continuously interacts with the network environment, learning optimal strategies in real-time, which allows it to detect previously unseen and evolving attack patterns more effectively. S. Dong et al. [17], their SSDDQN-based model for abnormal traffic detection showed improvements in detection accuracy through dimensionality reduction and unsupervised clustering. However, it primarily focuses on abnormal traffic detection without addressing adaptive modeling and novel attack detection. However, this paper's approach enhances adaptability by dynamically learning from the environment and using multi-dataset validation, ensuring better performance in detecting new threats and handling diverse network environments. R. R. d. Santos et al. [19] highlighted the challenges of evolving traffic patterns, requiring frequent updates in machine learning models. Our DRL-based approach eliminates the need for constant manual updates by learning from continuous interactions with the network, making it more resilient to evolving attacks and significantly reducing the resource demands for model maintenance. X. Ma and W. Shi [20]: combined RL with SMOTE to address class imbalance, but relied heavily on the NSL-KDD dataset, which may not generalize well to real-world scenarios. Author's work in this study's model goes beyond this by performing multi-dataset validation to ensure the robustness of our results across diverse network environments, making it applicable to real-world deployment. S. Yu et al. [25]'s DRL-based IDS for Industrial IoT faced scalability challenges in large deployments. Counter to that, this work overcomes this by ensuring scalability through efficient data balancing and adaptive modeling, making it suitable for both small-scale and large-scale network environments. This work's proposed DRL-based network intrusion classification addresses the limitations of prior works by offering dynamic adaptability, improved class imbalance handling, and real-time threat detection. Through multi-dataset validation, this solution is robust and reliable, ensuring that it can handle evolving and unseen threats across diverse network conditions. These advancements make this study a comprehensive and scalable solution for modern network security.

Chapter 3

Methodology

The methodology shown in fig. 3.1 is based on the development of DRL based Network Intrusion detection. Our methodology consists of 3 steps. Author has discussed each steps in detailed in following sub sections.

3.1 Data Collection

Author has collected dataset from open source NIDS datasets from University of Queensland [12] named NF-BoT-IoT, NF-ToN-IoT, NF-UNSW-NB15 and NF-UQ-NIDS. Each dataset consists of benign and attack data. Benign data are categorized as 0 and all types of attacks are categorized as 1. Our primary dataset is NF-BoT-IoT and rest of the datasets are used for multi-dataset validation task. NF-BoT-IoT consists of a total number of data flows of 600,100 out of which 586,241 (97.69%) are attack samples (Reconnaissance, DDoS, DoS, Theft) and 13,859 (2.31%) are benign. In NF-ToN-IoT, contains a total of 1,379,274 data flows, with 1,108,995 (80.4%) categorized as attack instances (Backdoor, DoS, DDoS, Injection, MITM, Password, Ransomware, Scanning, XSS) and 270,279 (19.6%) classified as benign. NF-UNSW-NB15 comprises 1,623,118 data flows in total, with 72,406 (4.46%) identified as attack samples (Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms) while the remaining 1,550,712 (95.54%) are classified as benign. The NF-UQ-NIDS dataset contains a total of 11,994,893 records, with 9,208,048 (76.77%) representing benign flows and 2,786,845 (23.23%) classified as attack instances (DDoS, Reconnaissance, Injection, DoS, Brute Force, Password, XSS, Infiltration, Exploits, Scanning, Fuzzers, Backdoor, Bot, Generic, Analysis, Theft, Shellcode, MITM, Worms, Ransomware). The LUFlow dataset [49] is a flow-based intrusion detection resource that offers a solid ground truth by integrating data with threat intelligence services. It includes telemetry data featuring emerging attack vectors, collected through the deployment of honeypots within the IP range of Lancaster University. LuFlow dataset consists of 31,329,479 samples, categorized into 30,659,274 benign and 670,205 attack samples.

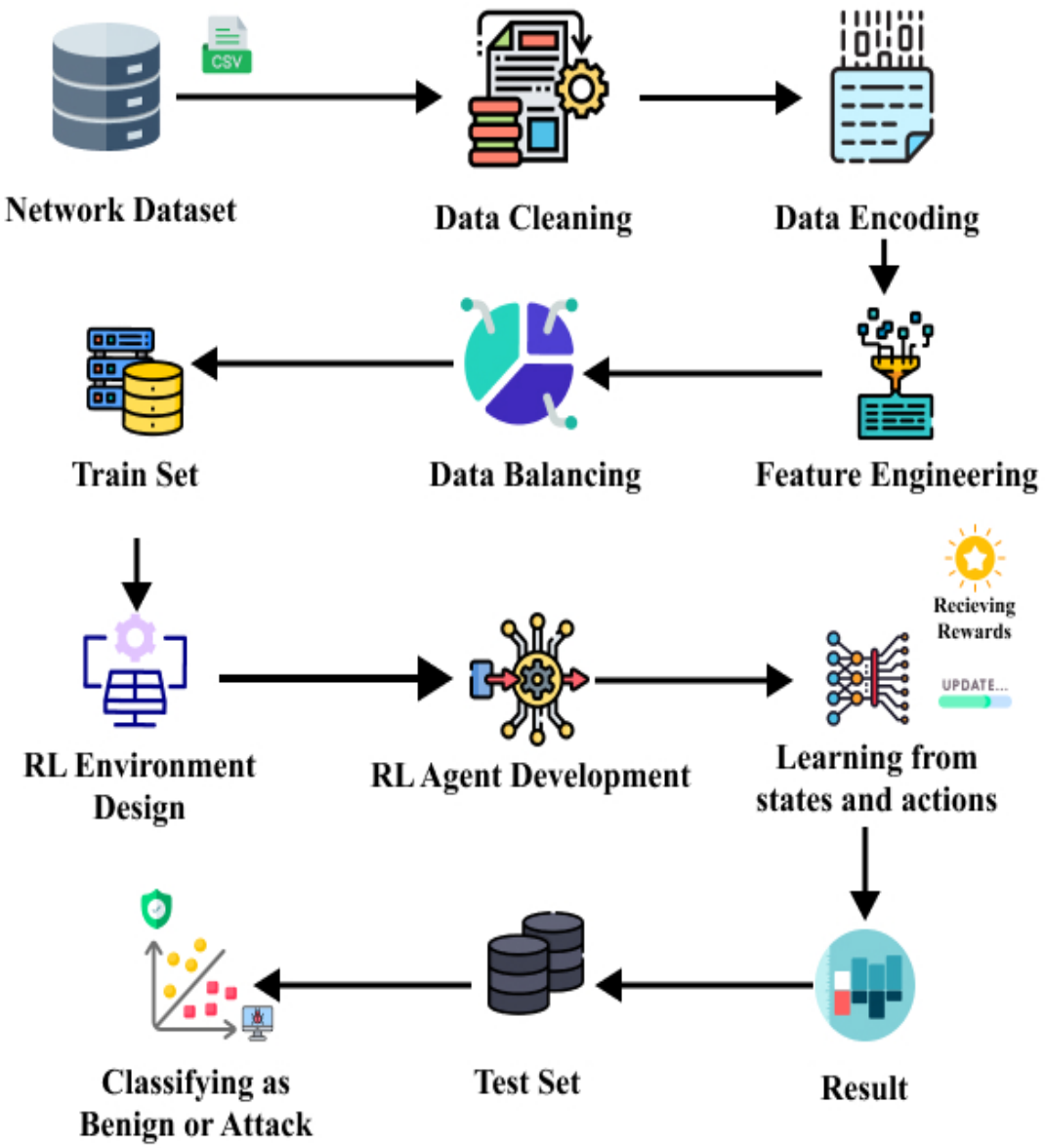


Figure 3.1: Diagram of the Proposed Methodology for a DRL Based Network Intrusion Detection System (NIDS)

3.2 Data pre-processing and Feature Engineering

Data preprocessing is a crucial step in any machine learning pipeline. It ensures that the data is clean, well-structured, and ready for analysis. In our work, the first step involves loading the dataset from a CSV file and checking for unnecessary columns. The IPV4_SRC_ADDR, IPV4_DST_ADDR, and Attack columns are dropped if they exist, as they may not contribute valuable information for the model, particularly we want to focus on raw network flow data. Next, categorical data is encoded. The LabelEncoder

transforms the categorical 'Label' column into numerical values, allowing the model to process these labels effectively. This step is important because many machine learning algorithms work with numerical data and cannot directly handle categorical variables. Missing values are then filled with zeros, ensuring that the dataset remains complete and preventing errors during model training.

After studying different datasets from literature, author has observed in our primary dataset these features should be included in order to make data more understandable for agent(proposed model) for accurate learning shown in Table II. Feature engineering enhances the dataset by creating new features that can provide additional insights and improve model performance. The code introduces several new features that are calculated based on existing data, which can help the model capture underlying patterns more effectively.

- Byte-to-Packet Ratios: The features IN BYTE PER_PKT and OUT_BYTE_PER_PKT are computed to reflect the average bytes per packet for incoming and outgoing traffic. This can help the model understand the efficiency of data transmission.
- Packet Rate per Flow Duration: The packet rate features (IN_PKT_RATE and OUT_PKT_RATE) indicate how many packets are transmitted per unit of flow duration, giving insight into network traffic intensity.
- Total Packets and Bytes: The features TOTAL_PKTS and TOTAL_BYTES summarize packets and bytes respectively coming in and going out. These summaries are critical for understanding the general activity of the network.
- Flow Duration per Packet: FLOW_DUR_PER_PKT explains the time taken per packet, which could be indicative of congestion or network performance issues.
- TCP Flag Features: Extracting flags such as TCP_SYN, TCP_ACK, and TCP_FIN helps the model understand the state of TCP connections, which is crucial for network behavior analysis.
- Binary Features for Common Protocols: Features like IS_DNS, IS_HTTP, and IS_HTTPS identify common Layer 7 protocols, allowing the model to focus on relevant traffic types.
- Interaction Features: The ratios IN_OUT_BYTES_RATIO and IN_OUT_PKTS_RATIO provide insights into the relationship between incoming and outgoing traffic.

These features added in this work not only enrich the dataset but also provide essential context for understanding network behavior. In the domain of network intrusion detection, being able to quantify and analyze these dimensions of network traffic is critical for: **Enhancing Detection Accuracy:** More informative features lead to better model performance by improving the classifier's ability to differentiate between benign and malicious

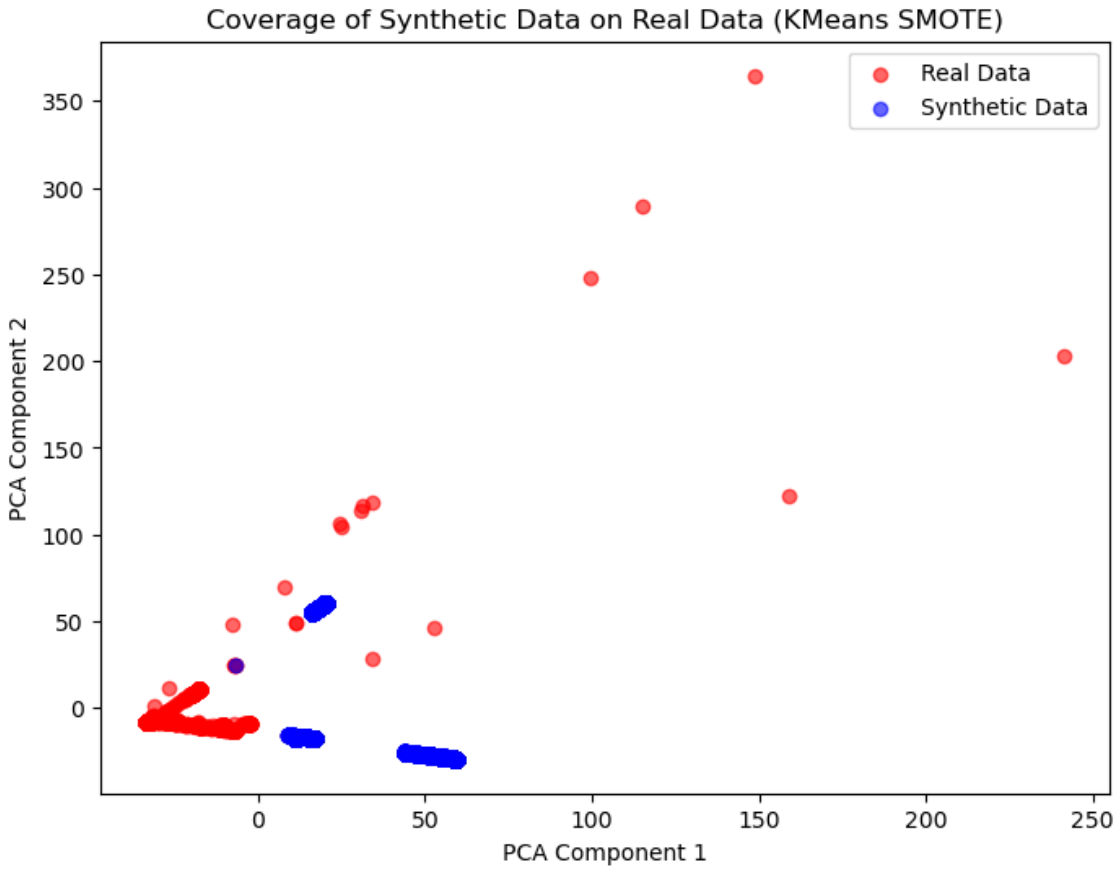


Figure 3.2: Coverage of synthetic data generated using KMeans SMOTE on the real data.

traffic.

Reducing False Positives: With a more nuanced understanding of typical network behavior, the model is less likely to misclassify benign activities as threats, which is vital for operational efficiency.

Improving Response Time: By identifying specific patterns associated with attacks, security teams can respond more swiftly and effectively to potential threats.

Each added feature serves to enhance the model's understanding of normal and abnormal network behavior, thus facilitating the accurate detection of intrusions and ensuring the security of networked systems.

Class imbalance is a common challenge in classification tasks, especially in areas like intrusion detection, where certain classes (e.g., benign traffic) may vastly outnumber others (e.g., different types of attacks). Author used Kmeans Smote for mitigating class imbalance issue. The use of KMeans SMOTE is particularly interesting, as it combines KMeans clustering with SMOTE to create more targeted synthetic samples. By clustering the data, the method generates synthetic samples that are representative of the minority class, thus enhancing the overall dataset's diversity and preventing overfitting. The class distribution after KMeans SMOTE is 586242 and 586241 for banign and attack. The fig. 4.3 shows a visualization of the coverage of synthetic data generated using KMeans SMOTE on the

Table 3.1: Feature Engineering Summary for Network Intrusion Classification.

Feature	**Description**	**Rationale**
IN_BYTE_PER_PKT	Average bytes per incoming packet.	Identifies the efficiency of incoming data transfers; lower values may indicate probing or abnormal traffic patterns.
OUT_BYTE_PER_PKT	Average bytes per outgoing packet.	Useful for detecting small, frequent packets that may suggest malicious activities.
IN_PKT_RATE	Rate of incoming packets per flow duration.	High rates can indicate scanning or DoS attacks, providing insights into abnormal network activity.
OUT_PKT_RATE	Rate of outgoing packets per flow duration.	Detects potential attack patterns where rapid outgoing packets may signify data exfiltration or other suspicious activities.
TOTAL_PKTS	Total number of packets (incoming + outgoing).	Sudden spikes can indicate ongoing attacks, allowing for quicker detection of unusual network behavior.
TOTAL_BYTES	Total number of bytes (incoming + outgoing).	Helps establish baseline network activity, aiding in the identification of anomalies indicative of intrusions.
FLOW_DUR_PER_PKT	Average flow duration per packet.	Low values may suggest rapid-fire attacks, helping to identify irregular timing patterns in packet transfers.
TCP_SYN	Indicator for SYN flag in TCP connections.	High SYN counts without corresponding ACKs may suggest SYN flood attacks, which are common in DoS scenarios.
TCP_ACK	Indicator for ACK flag in TCP connections.	Analyzing ACK patterns helps understand connection behaviors and detect potential connection anomalies.
TCP_FIN	Indicator for FIN flag in TCP connections.	Useful for assessing normal connection termination processes; irregular patterns may indicate malicious activity.
IS_DNS	Binary feature indicating whether traffic is DNS (port 53).	Critical for detecting DNS-related attacks, such as DNS amplification, allowing for targeted analysis of specific protocol behaviors.
IS_HTTP	Binary feature indicating whether traffic is HTTP (port 80).	Identifies traffic patterns that could signify HTTP-based attacks, aiding in the detection of vulnerabilities in web services.
IS_HTTPS	Binary feature indicating whether traffic is HTTPS (port 443).	Useful for assessing secure web traffic, where attacks may still occur despite encryption, enhancing detection capabilities.
IN_OUT_BYTES_RATIO	Ratio of incoming bytes to outgoing bytes.	Anomalies in this ratio can indicate potential data exfiltration or abnormal network interactions, helping to flag suspicious activity.
IN_OUT_PKTS_RATIO	Ratio of incoming packets to outgoing packets.	Helps assess the balance of traffic; significant imbalances may signal potential attacks or anomalies.

real data. The blue (synthetic) points are primarily located around clusters of red points, which shows that KMeans SMOTE has generated synthetic data near the real data points. This helps balance the class distribution by filling in areas where the minority class is underrepresented. This synthetic data helps balance the classes by adding data points for the minority class, reducing the skewness in the dataset. This makes it easier for machine learning models to learn and detect minority class instances (e.g., rare intrusions). The placement of synthetic data (blue points) closely follows the distribution of the real data (red points), as evidenced by the clustering around existing real data points. This shows that KMeans SMOTE carefully generates samples that are not random but informed by the structure of the existing data, preventing distortion of the overall dataset. After applying KMeans SMOTE, the dataset is split into training and testing sets using train_test_split. This ensures that the model is evaluated on a separate set of data, which is crucial for assessing its performance reliably. The new class distribution is printed to confirm that the imbalance has been effectively mitigated.

3.3 Proposed Deep Reinforcement Learning Framework

In this work, DRL-based NIC is developed using a hybrid deep learning architecture that combines a DNN and LSTM layers. The DNN+LSTM architecture is used as the Agent in the DRL framework, where the agent is trained to detect intrusions by interacting with network data in a sequential decision-making process. The following are the key components and steps of the system:

3.3.1 Problem Setup and Parameters

The problem is framed as a binary classification task (action size = 2), where the system decides whether the network activity is normal or malicious. The state size is determined by the number of features (x_train.shape[1]) extracted from the network traffic data. The agent uses the following key hyperparameters:
Discount Factor: 0.96, which helps to balance immediate and future rewards.
Epsilon: Initial exploration rate is set to 1.0, decaying over time to encourage exploitation of learned policies. Minimum epsilon is 0.07.
Learning Rate: Set at 0.0001, controlling the speed at which the model adjusts its weights.
Batch Size: 16, used during experience replay.
Episodes: The training process runs for 101 episodes.

3.3.2 Agent Design and Development

The agent design involves developing a hybrid architecture that integrates DNN and LSTM layers shown in Table 3.2. Below is a detailed explanation of each component of the architecture along with the relevant formulas. The DNN component of our model comprises

Dense, dropout and Activation layers. Dense layers are typically utilized to convert input training data into a high-dimensional feature space, facilitating the effective learning of complex features and underlying patterns within the network data. Additionally, the dense layer performs a linear transformation on the input data by multiplying it by a weight and incorporating a bias, as illustrated in equation (1), where w is the weight, x is the input data, and b is the bias. We have used ReLU activation layers with Dense. ReLU activation introduces sparsity in the network. Since neurons can output zero, this leads to a sparse representation, reducing the number of active neurons and potentially improving model efficiency and interpretability shown in eqn (2).

$$y = x_n \cdot w_n + b_n \tag{3.1}$$

$$ReLU(y) = max(0, y) \tag{3.2}$$

LSTMs are particularly suited for NIDS because they can effectively process time-series data, which is critical for analyzing network traffic patterns. In DRL-based NIDS, the LSTM layers contribute to the agent's ability to learn optimal policies for classifying network states as either benign or attack. The LSTM's unique architecture allows it to maintain relevant information from past observations, which is essential for making informed decisions based on the current state of network traffic. An LSTM cell consists of three main components: cell state, hidden state, and gates (Forget Gate,Input Gate) which can be calculated by following these equations (3-9).

Forget gate:

$$f_t = \sigma\left(w_{fh} \cdot h_{t-1} + w_{fx} \cdot x_t + b_f\right) \tag{3.3}$$

Input Gate:

$$i_t = \sigma\left(w_{ih} \cdot h_{t-1} + w_{ix} \cdot x_t + b_i\right) \tag{3.4}$$

Input Node:

$$g_t = \tanh\left(w_{gh} \cdot h_{t-1} + w_{gx} \cdot x_t + b_g\right) \tag{3.5}$$

$$c_{ti} = i_t \cdot g_t \tag{3.6}$$

$$c_t = c_{ti} + c_{tf} \tag{3.7}$$

Output gate:

$$o_t = \sigma\left(w_{oh} \cdot h_{t-1} + w_{ox} \cdot x_t + b_o\right) \tag{3.8}$$

Table 3.2: DRL Agent Architecture.

Layer (type)	**Output Shape**	**Activation Function**
Input Layer	(batch size, 25)	-
Dense Layer 1 (Dense)	(batch size, 128)	ReLU
Dropout Layer 1 (Dropout)	(batch size, 128)	-
Reshape Layer (Reshape)	(batch size, 1, 128)	-
LSTM Layer 1 (LSTM)	(batch size, 1, 64)	-
LSTM Layer 2 (LSTM)	(batch size, 1, 64)	-
LSTM Layer 3 (LSTM)	(batch size, 64)	-
Dense Layer 2 (Dense)	(batch size, 200)	ReLU
Dropout Layer 2 (Dropout)	(batch size, 200)	-
Dense Layer 3 (Dense)	(batch size, 150)	ReLU
Dropout Layer 3 (Dropout)	(batch size, 150)	-
Dense Layer 4 (Dense)	(batch size, 128)	ReLU
Dense Layer 5 (Dense)	(batch size, 32)	ReLU
Dense Output Layer (Dense)	(batch size, 2)	Softmax

$$h_t = \tanh(c_t) \cdot o_t \tag{3.9}$$

LSTM's process is characterized by selective memory enhancement. It starts with the current cell state, denoted as c_t. To determine this new cell state, the LSTM considers various factors, including the input states x_t and the previous hidden state h_{t-1}, which represent the incoming network traffic data and context from the previous time step. Crucially, the LSTM leverages a forget gate that outputs c_{ti} and a corresponding forget gate signal f_t. These components enable the LSTM to decide which information from the previous cell state should be retained and which should be discarded, thereby allowing the agent to filter out irrelevant network traffic data that may not contribute to intrusion detection.

By combining c_{ti} and f_t, the next cell state c_t is obtained. Additionally, the LSTM calculates an input gate i_t and an input node g_t using sigmoid and tanh activation functions, respectively, to determine what new information should be added to the cell state. This process is essential for the NIDS agent to incorporate new patterns of network behavior as they emerge. The output state o_t is derived by considering the inputs x_t and h_{t-1}, followed by the next hidden state h_t, which is generated by applying a tanh activation function to the current cell state c_t and combining it with the output gate o_t. This process allows the LSTM to selectively update its memory, fine-tuning the cell state while maintaining the context of past network activity. This selective updating is crucial for enhancing the agent's ability to accurately identify and respond to potential intrusions based on evolving patterns in network traffic. The final model is compiled with the categorical cross-entropy loss function, optimizing using the Adam optimizer, which updates the weights to minimize the loss:

$$Loss = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3.10}$$

where,

- C: The number of classes in the classification problem.
- yi: The true label for the i-th data point (1 if the i-th data point belongs to class c, 0 otherwise).
- ŷi: The predicted probability that the i-th data point belongs to class c.

3.3.3 Experience Replay

Experience Replay is a technique used to improve the sample efficiency and stability of the learning process in reinforcement learning. In this implementation, the DQNAgent class maintains a memory buffer (deque) called memory, which stores the agent's experiences as tuples of (state, action, reward, next state, done). This buffer has a maximum length of 1000, ensuring that older experiences are discarded as new ones are added. The update rule for Q-values is given by:

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a') \tag{3.11}$$

where:

- $Q(s, a)$ is the Q-value for state s and action a,
- r is the reward received for taking action a in state s,
- s' is the next state,
- γ is the discount factor, and
- $\max_{a'} Q(s', a')$ is the maximum Q-value for the next state s'.

The remember function Algorithm 3.1 appends new experiences to the memory buffer. This function is called after each action taken by the agent in the environment. While the agent interacts with the environment, it gathers varied experiences that can be reused in future learning cycles. During training, the agent samples experiences from the memory at random through the replay function in algorithm 3.2. Doing this step helps weaken the correlation between sequential experiences. In so doing, it allows the agent to generalize better on a wide range of situations and actions. For each experience in the minibatch, the agent computes the target value based on the reward received including the maximum predicted future reward from the next state.

This helps mitigate the problems caused by correlated observations and allows for a reuse of experiences in order to stabilize and improve the training process. At each

training, a mini-batch of experiences is randomly sampled from the memory buffer, and experiences are used to update the Q-values. The random sampling causes experiences to deviate from each other, which makes it more effective at learning.

Algorithm 3.1: Remember Function

Define experience tuple experience = (state, action, reward, next_state, done)
Append experience to memory: memory.APPEND(experience) =0

Algorithm 3.2: Replay Function

if length(memory) > batch_size **then**
  Sample a minibatch of experiences:
  minibatch ∼ RANDOM_SAMPLE(memory, batch_size)
  for each experience ∈ minibatch **do**
    Extract components: (state, action, reward, next_state, done) ← experience
    Reshape state: state ← RESHAPE(state)
    Reshape next state: next_state ← RESHAPE(next_state)
    Compute target: target ← reward
    if NOT done **then**
      Update target: target ← reward + $\gamma \cdot$ MAX(PREDICT(next_state, model))
    end if
    Predict current state: target_f ← PREDICT(state, model)
    Update prediction for action: target_f[action] ← target
    Train model: FIT_MODEL(state, target_f, model, class_weights)
  end for
  if epsilon > epsilon_min **then**
    Decay exploration rate: epsilon ← epsilon · epsilon_decay
  end if
end if=0

3.3.4 Reward Mechanism

The reward mechanism serves as a critical component for learning and adapting the agent's behavior based on the actions it takes. The reward system in this work employs a simple yet effective binary feedback mechanism. Whenever the agent takes an action, which corresponds to classifying a particular input sample, it receives a reward based on the correctness of that action. Specifically, if the action corresponds to the true label of the input sample (y_train[time_step]), a reward of +1 is granted, as it suggests a correct classification; otherwise, a reward of -1 is given for misclassification. Such binary rewards basically stimulate the agent in improving his decision-making capabilities through the pattern identification that leads to correct classification. By using such a clear-cut reward assignment, the learning process can be interpretable and thus debug and understand how an agent interacts with the environment.

Discounted Rewards: In addition to these immediate rewards assigned for each action,

there is also the aspect of discounted rewards-what in the long run the agent can expect to get as a reward. An implementation that makes this possible is through the incorporation of a discount factor, in this case represented by the variable gamma and set at 0.96. The discounting factor is that which determines the relative weighting between future rewards and immediate rewards. During training, as the agent moves along the time steps, the discounted reward is estimated by adding the immediate rewards, considering the temporal importance of the latter. For example, the discounted reward for an action at a certain time step is estimated by multiplying the reward by the discount factor raised to the power of the time step. It helps the agent to learn not just from the immediate feedback but also from the future consequences of their actions, making the agent act more strategically.
Exploration vs. Exploitation: The reward mechanism is closely related to the trade-off between exploration and exploitation and is controlled by the epsilon-greedy strategy. The epsilon value, standing for the rate of exploration, was initialized to 1.0 to ensure intensive exploration of the action space. Later, these epsilon values decay slowly during training towards a minimum of 0.07 to encourage exploitation in learned actions. All this is deeply combined with the reward system, which is supposed to be of primary importance for the agent in its decisions-to explore new actions or to exploit the ones that are already known and yield higher rewards. It will improve the policy of balancing exploration and exploitation in classifying instances within the dataset.

3.4 Hardware Specification

Therefore, to implement the proposed DRL-based network intrusion classification effectively, some specific hardware requirements are shown in table 3.3. The detailed breakdown for the hardware used for experimentation is as follows:

Processor: This system was tested using an Intel Core i5-8265U CPU at 1.60 GHz base clock, with 8 logical cores. Thus, the computational power would suffice for the training and inference of DNN and LSTM models. The time taken for processing can be further brought down by the higher-range processors to improve real-time performance in larger network environments.

RAM: A total of 12 GB of RAM was utilized, which proved sufficient for managing network traffic data and deep learning model training. However, increasing the RAM would be beneficial for handling larger datasets or for parallel processing during model training and validation.

Graphics: The system was equipped with Intel® UHD Graphics 620, which includes 6197 MB of total graphics memory. Out of this, 128 MB is dedicated display memory, while 6069 MB is shared memory. Although integrated graphics were sufficient for the experiments conducted in this study, the use of dedicated GPUs (e.g., NVIDIA) with CUDA support is highly recommended for faster model training and handling of large-scale network intrusion datasets.

Table 3.3: Hardware Specification.

Component	**Specification**	**Remarks**
Processor	Intel® Core™ i5-8265U CPU @ 1.60GHz (8 logical cores)	Sufficient for model training and inference, though more powerful CPUs can reduce training time.
RAM	12 GB	Adequate for network traffic data and model training. More RAM is recommended for larger datasets.
Graphics	Intel® UHD Graphics 620	Integrated graphics were sufficient, but a dedicated GPU is recommended for better performance.
Total Graphics Memory	6197 MB	Total memory available for graphics processing, shared between system and display.
Display Memory	128 MB	Dedicated display memory for handling visual output.
Shared Memory	6069 MB	Shared between system and graphics tasks, reducing GPU burden.

Chapter 4

Results Analysis

After successful training process, author has conducted multi-dataset validation, Comparative Analysis on Synthetic Data balancing methods, used LIME for understanding the feature contribution and tested the Adaptive modeling functionality on our proposed model (agent). Author has used performance matrix such as accuracy, f1-score, recall, precision, AUC-ROC, total reward, average reward, discounted reward and confusion matrix for evaluating the proposed solution shown in table 4.3 .

4.1 Performance Analysis

In table 4.1, 4.2 the classification report of both Training and Testing of network intrusion classifier.

Table 4.1: Classification Report on TRAINING data at episode 100 (Weighted), Dataset:NF-BoT-IoT.

Class	**Precision**	**Recall**	**F1 Score**	**Support**
0	1.00	0.98	0.99	468736
1	0.98	1.00	0.99	469250
ACCURACY			**0.99**	**937986**
MACRO AVG	**0.99**	**0.99**	**0.99**	**937986**
WEIGHTED AVG	**0.99**	**0.99**	**0.99**	**937986**

Convergence Rate indicates the rate at which the learning algorithm converges to an optimal policy. A convergence rate of 0.1200 suggests that, at episode 100, the learning process is stabilizing. Total reward is the sum of all rewards received by the agent throughout the episodes. A higher total reward indicates that the agent is performing well in terms of the objectives set for it, effectively learning to maximize rewards through its actions. The discounted reward accounts for future rewards, providing a present value to rewards received in the future. The discount factor adjusts the importance of future rewards. A value of 20.56 suggests that the agent is optimizing for both immediate and

Table 4.2: Classification Report on TESTING data at episode 100 (Weighted), Dataset:NF-BoT-IoT.

Class	**Precision**	**Recall**	**F1 Score**	**Support**
0	1.00	0.98	0.99	117506
1	0.98	1.00	0.99	116991
ACCURACY			**0.99**	**234497**
MACRO AVG	**0.99**	**0.99**	**0.99**	**234497**
WEIGHTED AVG	**0.99**	**0.99**	**0.99**	**234497**

Table 4.3: Performance Metrics of DRL-based Network Classifier, Dataset:NF-BoT-IoT.

Metric	**Train Data**	**Test Data**
AUC-ROC	0.9896	0.9895
Accuracy	0.9896	0.9895
Precision	0.9797	0.9794
Recall	1.0000	1.0000
F1-Score	0.9898	0.9896

Metric	**Score**
Convergence Rate	0.1200
Total Reward	46
Average Reward	0.92
Discounted Reward	20.56

future outcomes effectively.

While dealing with class imbalance issue author has used several methods like SMOTE-ENN, ADYSN, Kmeans SMOTE and Borderline SMOTE. The impact of choosing right method for accurate intrusion detection is important. A set of experiment conducted with following synthetic data generation methods shown in table 4.4. Kmeans SMOTE consistently outperformed the other methods across all metrics, making it the most effective approach in handling class imbalance for network intrusion detection. Meanwhile, SMOTE-ENN had the lowest performance, suggesting it might not be the ideal choice in this context.

Table 4.4: Performance Metrics of Different Synthetic Data Methods.

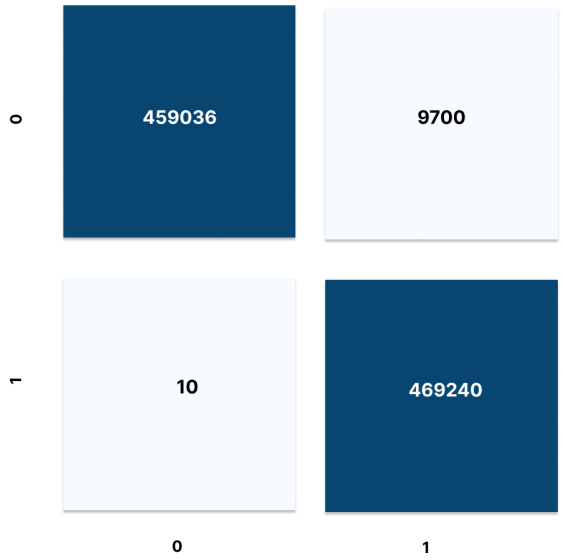
Method	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC-ROC**
Kmeans SMOTE	0.9895	0.9794	1.0000	0.9896	0.9895
ADYSN	0.9452	0.9643	0.9246	0.9440	0.9452
Borderline-SMOTE	0.9458	0.9682	0.9216	0.9443	0.9457
SMOTE-ENN	0.8817	0.9116	0.8476	0.8784	0.8820

A confusion matrix is a key tool for evaluating the performance of a classification model, such as those used in NIC. It provides a detailed breakdown of how well the model distinguishes between different classes—in this case, whether network traffic is normal or attack. The matrix organizes classification outcomes into four categories: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). If an attack is detected and it truly is malicious traffic, this is counted as a true positive. False positives occur when the system incorrectly classifies normal network traffic as an intrusion. This is also known as a false alarm. True negatives represent instances where the system correctly identifies normal traffic as non-intrusive. False negatives occur when the system fails to detect an actual intrusion, incorrectly classifying it as normal traffic. In fig. 4.1 author presented the proposed model's confusion matrix report.

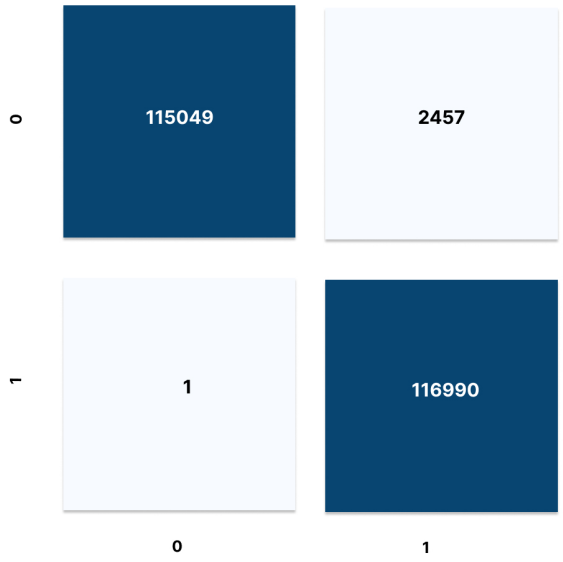
Table 4.5: Performance Evaluation of the effectiveness of Our approach Across multiple Datasets.

Dataset	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC-ROC**
NF-BoT-IoT	0.9895	0.9794	1.0000	0.9896	0.9895
LUFlow	0.9228	0.9143	0.9346	0.9244	0.9227
NF-UNSW-NB15	0.9768	1.0000	0.9536	0.9762	0.9768
NF-UQ-NIDS	0.9486	0.9889	0.9074	0.9464	0.9486
NF-ToN-IoT	0.9423	0.9273	0.9600	0.9434	0.9423

In order to prove the robustness of the proposed approach across various types of datasets, author has conducted a multi-dataset validation strategy, as shown in Table 4.5. Additionally, author performed a comparative analysis of the proposed model against prior studies, which is illustrated in Table 4.6. This comparison evaluates the performance of established models, such as GRU + LSTM, CNN + LSTM, and DNN, against the results obtained from the proposed model. The evaluation is based on five key metrics: Accuracy, Precision, Recall, F1 Score, and AUC-ROC. Our proposed model demonstrates superior performance, achieving an Accuracy of 0.9895 and an AUC-ROC of 0.9895, indicating exceptional overall detection capability. The Recall of 1.00 means that it correctly classified all actual intrusions. The model still sustains a high Precision of 0.9794, indicative of a low false positive rate, which has a very important consequence in minimizing false alerts. The F1 Score was 0.9896, which further underscores the high performance of this model, hence showing a remarkable balance between Precision and Recall. This balance underlines the model's capability for accurate and reliable classification of network intrusions, making it a strong solution in the cybersecurity landscape. The results highlight the effectiveness of the proposed approach, affirming its potential as a leading model in the domain of network



(a) Confusion Matrix on Train set



(b) Confusion Matrix on Test set

Figure 4.1: Confusion Matrix on the NF-BoT-IoT Dataset.

intrusion classification.

Additionally, author has presented confusion matrix of our model in different dataset which was used for multi-dataset validation shown in fig. 4.2.

Table 4.6: Prior study comparison, Dataset: NF-BoT-IoT.

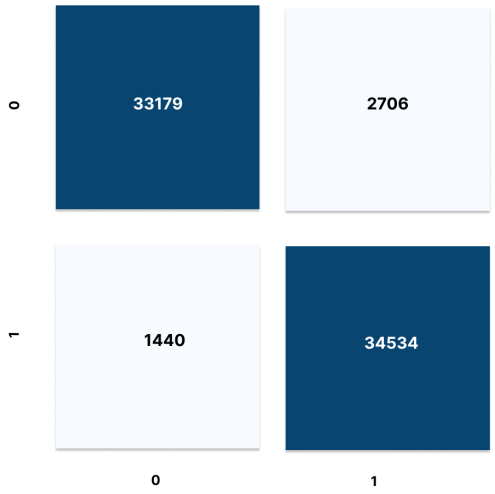
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC-ROC**
GRU + LSTM [50]	0.9678	0.9940	0.9411	0.9668	0.9677
CNN + LSTM [51, 52]	0.9556	0.9523	0.9346	0.9534	0.9556
DNN [53]	0.8845	0.9961	0.7715	0.8696	0.8843
Our Work	0.9895	0.9794	1.0000	0.9896	0.9895

4.2 Interpreting Feature Contributions with LIME

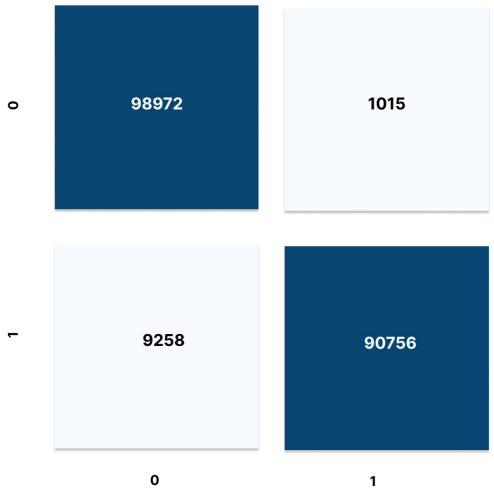
LIME, or Local Interpretable Model-agnostic Explanations, is a powerful technique that has been developed to enhance the interpretability of "black-box" machine learning models. The term "black-box" usually refers to deep learning networks or ensemble methods that, though efficient, are typically criticized for their lack of transparency when it comes to decision-making. This may be a cause of alarm for many complex models because, by their very nature, it's hard to understand how they make a particular prediction-especially major applications that range from health, financial, and legal systems where trust and accountability are paramount. That is where LIME addresses this issue by supplying a way to explain the predictions for those complex models on an individual basis. LIME works by locally approximating the black-box model-that is, it creates an interpretable, simpler model, such as a linear model, around the prediction for a certain instance or data point. In this way, and focusing on one prediction at a time, LIME is able to pinpoint which of the different features or inputs contributed toward that one particular outcome. This can be explained locally; thus, it goes a long way in making it easier for users to explain why a model has made a certain decision by taking into consideration the feature importance or influence that is used in the model to make a prediction.

A main advantage of LIME is that it is model-agnostic, which in principle enables the explanation of any machine learning model regardless of the underlying structure or complexity. Whether the model is a simple decision tree, or a complex deep neural network, or even an ensemble like a random forest, LIME can enable users to develop a better understanding of how the model works at the level of individual predictions. This makes LIME a very important tool in enhancing the transparency and accountability of machine learning systems, particularly in situations where explaining automated decisions is imperative. By providing these local explanations, LIME helps to create trust in machine learning models, thus making them more comprehensible and accessible to a wide variety of stakeholders who could either be technical or nontechnical.

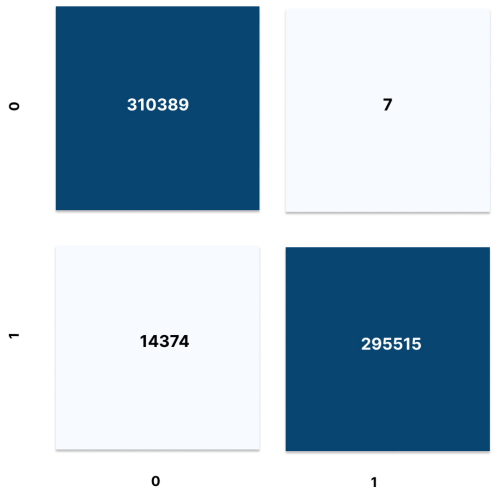
The LIME output visualization shown in fig. 4.3 provides a detailed breakdown of how the network intrusion classification model arrived at its prediction for a specific instance.



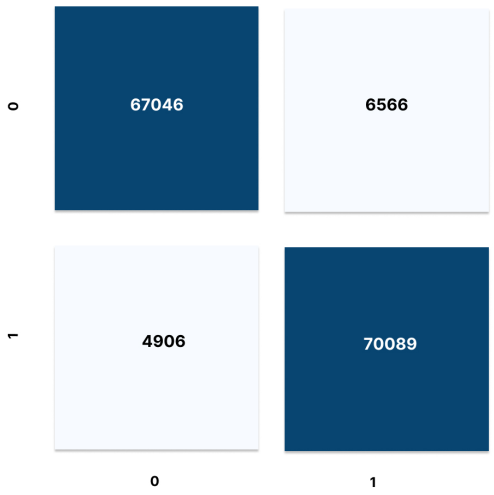
(a) Confusion Matrix, Dataset=NF-ToN-IoT



(b) Confusion Matrix Dataset=NF-UQ-NIDS



(c) Confusion Matrix Dataset=NF-UNSW-NB15



(d) Confusion Matrix Dataset=LuFlow

Figure 4.2: Confusion Matrix of the proposed model in different network Dataset.

In this case, the model predicted that the traffic belongs to the "Benign" class with a probability of 1.00 (or 100%), while assigning a probability of 0.00 (or 0%) to the "Attack" class. This clear prediction confidence is crucial for understanding how certain the model is about its decision. The contribution of individual features to the prediction is shown through a vertical bar chart. Each feature is represented by a bar that either pushes the prediction towards the "Benign" class (on the left, shown in blue) or towards the "Attack" class (on the right, shown in orange). The length of the bar indicates the strength of that feature's influence on the prediction: longer bars represent features that have a greater impact, either positively or negatively, on the classification.

On the benign side, features such as "IS_HTTP $>$ 26.04" and "IS_HTTPS $>$ 31.55"

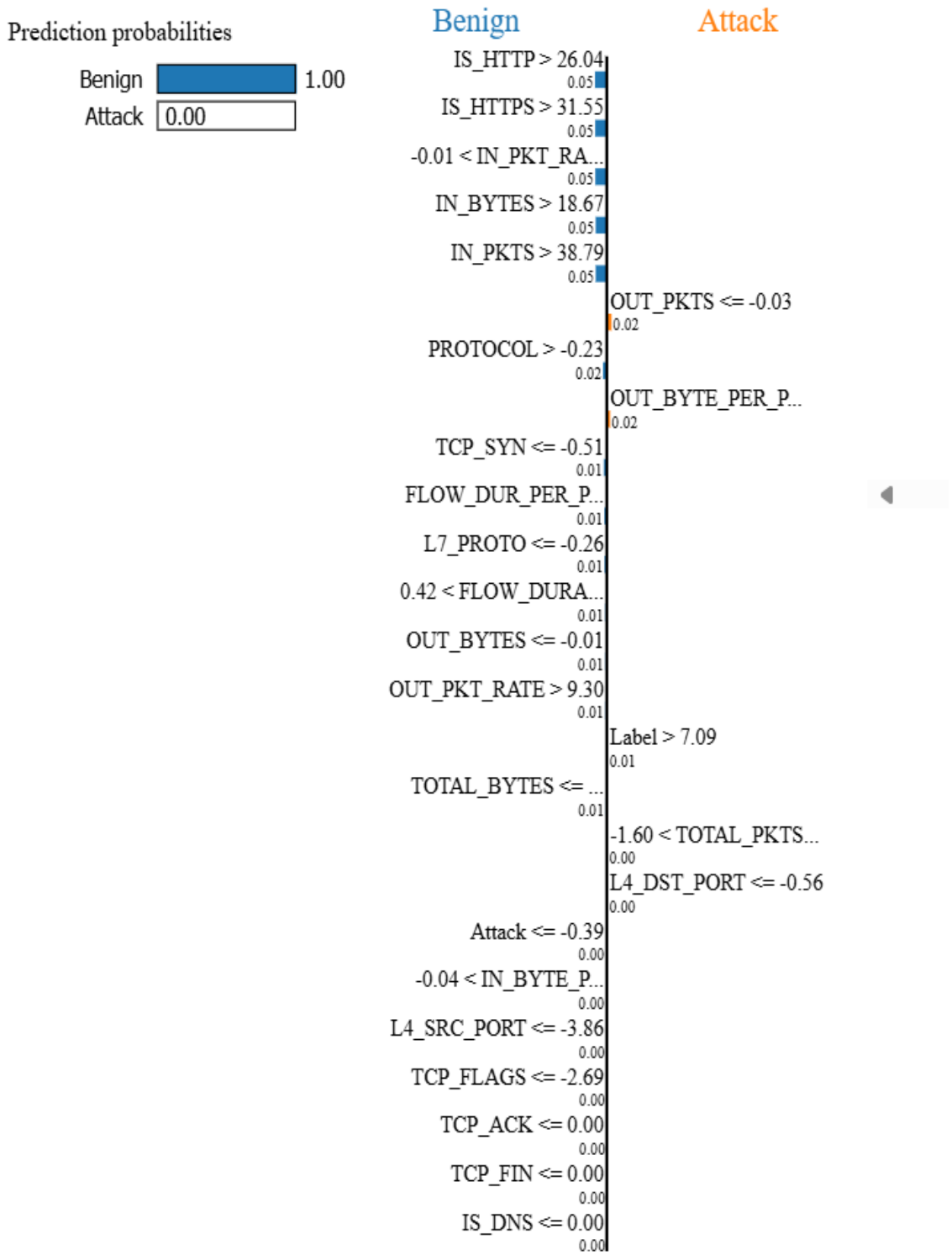


Figure 4.3: Feature contribution visualization using LIME

are important indicators. These features suggest that the traffic is likely composed of HTTP or HTTPS requests, which are common in legitimate web traffic. Therefore, their presence strongly pushes the prediction towards benign behavior. Similarly, features like "IN_BYTES > 18.67" and "IN_PKTS > 38.79," which indicate a larger volume of incoming

bytes and packets, suggest typical benign activities like downloading data or browsing, further reinforcing the benign classification. Another significant feature, "PROTOCOL <= -0.23," refers to the protocol being used, likely pointing to a common, benign protocol like TCP or UDP, which is associated with legitimate traffic. Additionally, features related to flow duration and total bytes, such as "FLOW_DUR_PER_P" and "TOTAL_BYTES," further contribute to the benign prediction by indicating normal, sustained communication patterns.

Features like "OUT_PKTS <= -0.03" and "OUT_BYTE_PER_P" on the attack side indicate lower outgoing traffic or bytes per packet. In many types of network attacks, such as Denial of Service, there is usually an incoming flood while very little data goes out; these features would therefore push the prediction to an attack side. Then, the feature "OUT_PKT_RATE > 9.30" indicates that, just like the incoming packet rate, the outgoing packet rate is very high, which could be indicative of an adverse process trying to exfiltrate data or flood the network and thus drive the model to consider the traffic as suspicious. This may also be indicative of the known patterns or labels of previous attacks, which makes this feature important.

These features are important because they show an explanation of the reasoning behind the prediction made by this model. The LIME output helps to demystify the "black-box" nature of machine learning models by showing exactly which features were most important in driving the benign prediction. Features like IS_HTTP, IN_BYTES, and IN_PKTS suggest normal web activity, giving a clear rationale for the benign classification. In contrast, the features related to outgoing packets and packet rates indicate abnormal traffic behaviors, which may be related to the attacks. This interpretability level is of high importance to network administrators, as it may provide them with a deeper understanding of how the model makes decisions. The understanding of the reasons behind the model's prediction enables the administrators to give appropriate responses to security alerts with a better insight, thus avoiding flagging of benign traffic and accurately detecting potential attacks. This also helps in identifying any deficiency in the model that may include features over-reliance, which can result in biased or incorrect predictions.

Chapter 5

Adaptive Modeling Evaluation

For testing the functionality of the adaptive modeling, author utilized the NF-BoT-IoT dataset. This dataset categorizes data points into two main groups: 'Benign', labeled as 0, and 'Attack', labeled as 1. Within the attack category, there are multiple specific types of attacks, including Reconnaissance, Distributed Denial of Service (DDoS), Denial of Service (DoS), and Theft. To improve the robustness of the model, author applied additional feature engineering as described in chapter IV-B. After incorporating these new features, author made an intentional adjustment to the dataset. Specifically, author excluded the 'DoS' attack samples from the training dataset and reserved them solely for the testing phase. This allowed me to evaluate the model's ability to generalize and detect unseen attacks during testing. After conducting experiment, author has seen the model performed better in detecting unknown attack (DoS is unknown for training data in this case).

5.1 Adaptive Modeling Results

Table 5.1: Adaptive Modeling performance: Classification Report (Weighted), Dataset:NF-BoT-IoT.

Class	Precision	Recall	F1 Score	Support
0	0.98	0.98	0.98	117506
1	0.98	0.98	0.98	116991
ACCURACY			**0.98**	**234497**
MACRO AVG	**0.98**	**0.98**	**0.98**	**234497**
WEIGHTED AVG	**0.98**	**0.98**	**0.98**	**234497**

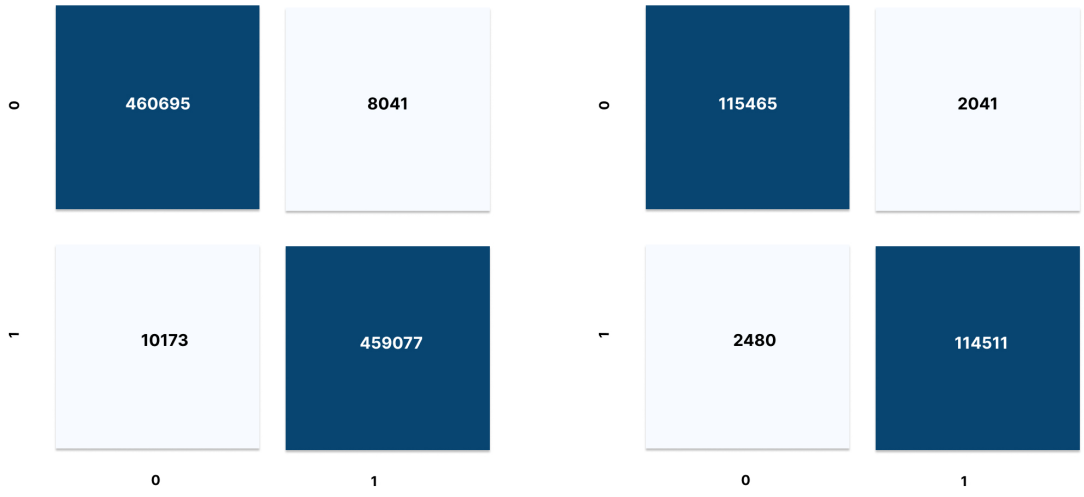
The results shown in table 5.1, 5.2 from the classification report highlight the effectiveness of the adaptive modeling approach applied in this experiment, specifically after excluding certain attack types (e.g., DoS) from the training data. The model was trained without seeing any 'DoS' attack samples, which were reserved solely for testing. Despite

this, the high performance across all key metrics—precision, recall, F1-score, and accuracy—indicates the model's ability to generalize well and detect unseen attack types.

The classification report in fig. 5.1(b) shows both precision and recall of 0.98 for 'Attack' class (label 1), which includes the previously unseen DoS attacks. This suggests that the model, though never explicitly trained on DoS samples, was able to recognize them during testing. This high recall means the model identified nearly all attacks correctly, including those not part of its training data, showcasing the adaptability of the model to new, unknown threats. The confusion matrix further confirms the adaptive performance, with 115,465 correct predictions for benign samples and 114,511 for attack samples. The misclassification rate is very low, as only 2,041 benign instances were classified as attacks and 2,480 attack instances as benign. These small numbers of misclassifications, compared to the total number of instances, demonstrate the model's capacity to handle new and unseen data effectively. The adaptive modeling has allowed the system to generalize beyond the training data, accurately detecting attacks it had not previously encountered, like DoS. This adaptability is crucial in cybersecurity, where new and evolving threats are constantly emerging.

Table 5.2: Adaptive Modeling Performance (Train and Test) , Dataset:NF-BoT-IoT.

Metric	**Train Data**	**Test Data**
AUC-ROC	0.9807	0.9895
Accuracy	0.9807	0.9895
Precision	0.9828	0.9825
Recall	0.9783	0.9788
F1-Score	0.9805	0.9806



(a) Confusion Matrix on Train set

(b) Confusion Matrix on Test set

Figure 5.1: Adaptive Modeling Performance: Confusion Matrix

5.2 Scenario: Adaptive Threat Detection in a Corporate Network

Let's consider a scenario in a large corporate environment, where a multinational company manages thousands of interconnected devices, including employee workstations, servers, IoT devices, and mobile endpoints. The company has a high volume of network traffic due to remote work, cloud services, and partner integrations, making it a prime target for cyber threats such as DDoS attacks, malware infiltration, or data ex filtration attempts.

In this scenario, the company's security team uses traditional signature-based IDS. These systems effectively detect known attacks but struggle with identifying novel or evolving threats. For instance, if an advanced persistent threat (APT) actor deploys a new variant of malware or initiates a low-frequency DDoS attack (slow-loris), the static IDS might miss the attack until significant damage is done.
Key Challenges:

- Dynamic Threats: Attackers continuously change tactics, techniques, and procedures (TTPs), making it difficult for static models or signature-based systems to keep up.
- High Volume of Traffic: With the vast amount of network traffic flowing through the corporate network, detecting low-frequency or low-signature attacks (such as slow-loris DDoS or stealthy data exfiltration) becomes challenging.
- Class Imbalance: The network traffic data is heavily skewed towards normal behavior, making it harder for the system to detect rare but critical attack instances.

5.2.1 Advantages of Proposed DRL-Based Approach

In this setting, the proposed DRL approach combined with a hybrid DNN-LSTM model offers several advantages over traditional systems:

- Real-Time Detection of Unseen Attacks: DRL continuously learns and adapts to changes in the network environment. For example, if an attacker launches a previously unknown form of data ex filtration or a new variant of ransomware, the DRL model can adapt by learning from interactions and feedback, adjusting its detection strategies to flag suspicious behavior even when the threat is not part of the pre-trained data.
- Adaptation to Evolving Attack Strategies: By excluding certain attack types (like DDoS) from the training phase but including them in the testing phase, the model demonstrates an ability to generalize and adapt. If the corporate network starts facing new attack vectors (like a slow-loris DDoS attack that gradually consumes server resources), our DRL-based system would be able to detect the evolving nature of the attack and prevent the disruption before it becomes catastrophic.

- Handling Imbalanced Data: Network traffic in such environments is overwhelmingly normal with rare but critical intrusion attempts. our approach addresses this by using advanced data balancing techniques like K-means SMOTE. In this scenario, the imbalanced network data (i.e., large volumes of legitimate traffic with few intrusion samples) is handled more effectively, ensuring the model doesn't miss subtle yet critical threats.

- Robust Multi-Dataset Validation: Since the model has been validated across multiple datasets, it can be confidently deployed in environments with different types of network flow data, such as cloud traffic, VPN connections, or IoT communications within the corporate network. This multi-dataset validation ensures that the DRL-based system is resilient to variations in traffic patterns, making it versatile across different sectors (e.g., banking, healthcare, or manufacturing).

5.2.2 Scenario Outcomes

The DRL-based model quickly detects an anomaly: the employee workstation communicates with an external IP address that has never been used before. Traditional systems, using known signatures, would not be able to flag this activity as malicious. This model learned from the event and labeled the behavior as possibly malicious since the pattern of low-volume, periodic transmission of data fits within the paradigm of attempts at data exfiltration. It isolates the device and notifies the security team about the presence of an APT actor who tries to extract sensitive corporate data. In addition, it observes a gradual traffic build-up toward a critical server and recognizes that a low-frequency DDoS attack is in effect. It deals with the situation by throttling the suspicious traffic, thus avoiding server downtime. In this regard, our proposed network intrusion classifier, based on DRL, can perform much-enhanced protection adapted to learning and robust against the imbalance of datasets. Traditional methods are usually at their best in detecting and responding to novel and evolving threats in real-time, which becomes quite critical for securing complex and high-traffic corporate networks.

5.3 Need for Advanced Learning Depth

Traditional deep learning methods for network intrusion classification train models to recognize attack patterns within historical data. While these methods can ensure good detection rates, they are inherently static and cannot adapt to new, previously unknown attack types evolving over time. An intrinsic limitation with such a model is that it relies on the patterns seen during training. The optimization of computational cost might be at the cost of adaptability, and these models probably miss such an anomaly when they face an attack that differs from learned patterns. That brings us to DRL in the proposed model, which forms an adaptive learning mechanism. The key difference is that DRL dynamically interacts with the environment to learn optimal detection strategies over time rather than

merely relying on predefined attack signatures. This continuous learning aspect lets the DRL-based models detect those patterns of attacks that previously could not be seen or were novel, commonly referred to as zero-shot attacks, wherein an attack is detected though it was not present in the training data. The Adaptive Modeling Experiment in chapter 5 acts as empirical evidence for this claim. In the experiment, certain kinds of attacks were excluded in the training set but added into the test set; say, the Dos type of attack. Whenever this happens, traditional deep learning methods often cannot detect those novel attacks because their detection capability tightly depends on the pattern that they have learned during the training process.

However, our DRL-based approach demonstrates superior performance under these conditions. By interacting with the evolving attack patterns in the test data and continuously refining its strategies, the DRL model shows a marked improvement in detecting these excluded attack types. This ability to generalize and adapt, even in the face of unseen attacks, provides a significant advantage over traditional static models. It ensures that the model can function in real-time network environments, where attacks are constantly evolving and attackers often introduce new techniques that deviate from historical trends. Thus, while traditional deep learning methods may come with lower computational costs, they sacrifice the crucial ability to adapt in real-time, limiting their effectiveness against zero-day or novel attacks. DRL's adaptive modeling, as evidenced by our zero-shot attack detection experiment, justifies the added complexity, as it enhances security in dynamic and unpredictable network environments, providing long-term robustness that traditional methods lack.

Chapter 6

Conclusion

The proposed deep reinforcement learning-based network intrusion classification framework demonstrates a significant advancement over traditional static models by incorporating a hybrid architecture of Deep Neural Networks and Long Short-Term Memory networks. The dynamic adaptability of DRL enables real-time detection of novel and evolving threats, which is critical in today's rapidly changing network environments. By effectively addressing the class imbalance issues inherent in network intrusion datasets through the innovative use of K-means-based data balancing, author has achieved superior performance across various evaluation metrics. The multi-dataset validation confirms the robustness of our approach, highlighting its potential as a reliable solution for enhancing the security of modern network infrastructures. Future work may explore further optimizations and the integration of additional machine learning techniques to further enhance detection capabilities.

6.1 Discussion

The feature engineering and dataset adjustments explained in chapter 3-B also contribute to the success of this adaptive modeling approach. In this case, by excluding the samples of the DoS attack type for training and saving only the test portion, the experiment has achieved, in fact, checking the model capability for adaptation against unseen data. The high performance, despite these constraints, gives an excellent reinforcement to its generalization. This flexibility is crucial in the dynamic cybersecurity domain for modeling not only the detection of known threats but also for finding out novel attack patterns, which might be used to exploit vulnerabilities. Besides, the total accuracy of 98% across the whole data space underlines the robustness of the model. The high accuracy of results obtained while testing the model against data with previously unseen types of attacks underlined the efficiency of the adaptive modeling strategy. That will keep the model agile and certain for both known and unknown threat detection in real time, which is one main requirement of intrusion detection systems operating in complex and ever-evolving network environments.

The result of this experiment underlines the potential of adaptive modeling within cybersecurity applications. With this high precision and recall against known and unknown attack types, the model is really an interesting approach to intrusion detection, one that's resilient. That is, such an approach will be able to generalize novel threats with very minimal performance loss, without constant retraining on new attack types, making it a more viable solution for real-time, scalable network security. Further optimizations might be done in future research for these aspects: an extension of the feature set or experiments with different attack categories may well improve the capabilities of the model.

6.2 Limitation

While the proposed DRL-based network intrusion detection system offers several advancements over existing solutions, it also has some limitations that should be acknowledged. The integration of deep reinforcement learning, hybrid DNN-LSTM architectures, and advanced data balancing techniques can lead to a complex implementation process. Organizations may require specialized knowledge and skills to effectively deploy and maintain the system. he training phase of deep reinforcement learning involves running numerous simulations and iterations to optimize the model's parameters. This requires substantial computational resources, including powerful GPUs or TPUs, which can be costly. Training deep learning models can take a significant amount of time, especially as the size and complexity of the dataset increase. This can lead to delays in model deployment and responsiveness to evolving threats. Future work may explore more efficient algorithms, distributed computing, or hardware acceleration techniques to mitigate these issues.

6.3 Future Work

The proposed network intrusion detection system based on deep reinforcement learning runs remarkably compared to the conventional static models. However, there is still further work that could be done to enhance performance. An important aspect will be optimization of the present model. We will study hyperparameter tuning and also architecture modification to have the best from the present hybrid architecture using DNN and LSTM networks. With systematic optimization of these parameters, we expect an enhancement of detection rates while reducing computational overheads, hence increasing efficiency in real-time applications.

Further, we will discuss how complementary machine learning techniques can be integrated, including ensemble learning methods and state-of-the-art anomaly detection algorithms. Their integration may add new dimensions to the security and enhance the ability of our system to detect complex attack patterns with minimal false alarms. The challenge thrown up by such advanced threats may, therefore, be better responded to by the holistic approach that may yield an intrusion detection system. Improvement in real-time adaptability of our model will also be a subject of concentration. We believe in implementing

mechanisms which will enable the systems to learn continuously from new data and user feedback, ensuring effectiveness against emerging threats, especially in dynamic network environments. This shall be very instrumental in maintaining security considering rapidly changing attack strategies.

Secondly, we shall check the possibility of transfer learning to see whether knowledge from one domain or dataset can be used to improve the performance of the other. In this regard, such a strategy may help much when the labeled data is insufficient, thus giving leeway to how our system adapts more promptly to new environments and enhances its detection capabilities across various contexts. Regarding the validation, while the existing multi-dataset strategy already attests to the robustness of our approach, in future work, we will apply our approach to more diverse datasets that should represent different network environments and attack vectors. Wider validation will help further establish the reliability and effectiveness of our model in real-life applications. We will also conduct user-centric evaluation necessary for refining our system. Feedback from the end-user will therefore be important in giving insight into just how practical and usable this intrusion detection system is, be it technically well or not, in providing access and actionable insight to network administrators.

References

[1] N. Borgioli, F. Aromolo, L. Thi Xuan Phan, and G. Buttazzo. A convolutional autoencoder architecture for robust network intrusion detection in embedded systems. *Journal of Systems Architecture*, 156:103283, November 2024.

[2] M. F. Monir and D. Pan. Exploiting a virtual load balancer with sdn-nfv framework. In *2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6, 2021.

[3] M. Shahin, M. Maghanaki, A. Hosseinzadeh, and F. F. Chen. Advancing network security in industrial iot: A deep dive into ai-enabled intrusion detection systems. *Advanced Engineering Informatics*, 62:102685, October 2024.

[4] M. F. Monir and F. Granelli. Exploiting wireless links diversity in software-defined ieee 802.11 enterprise wlan. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[5] T. Nandy, R. Md Noor, R. Kolandaisamy, M. Y. I. Idris, and S. Bhattacharyya. A review of security attacks and intrusion detection in the vehicular networks. *Journal of King Saud University - Computer and Information Sciences*, 36(2):101945, February 2024.

[6] D. Manivannan. Recent endeavors in machine learning-powered intrusion detection systems for the internet of things. *Journal of Network and Computer Applications*, 229:103925, September 2024.

[7] Y.-D. Lin, H.-X. Huang, D. Sudyana, and Y.-C. Lai. Ai for ai-based intrusion detection as a service: Reinforcement learning to configure models, tasks, and capacities. *Journal of Network and Computer Applications*, 229:103936, September 2024.

[8] J. F. Cevallos M., A. Rizzardi, S. Sicari, and A. Coen Porisini. Deep reinforcement learning for intrusion detection in internet of things: Best practices, lessons learnt, and open challenges. *Computer Networks*, 236:110016, November 2023.

[9] W. Z. T. Tareq and M. F. Amasyalı. Reinforcement learning algorithms. In *Decision-Making Models*, pages 339–350. Elsevier, 2024.

[10] C. Xie, R. Yao, Z. Liu, L. Zhu, and X. Chen. Process performance prediction based on spatial and temporal feature extraction through bidirectional lstm. In *Computer Aided Chemical Engineering*, pages 1615–1620. Elsevier, 2022.

[11] R. E. Thomson and W. J. Emery. Neural networks, convolutional neural networks and deep learning. In *Data Analysis Methods in Physical Oceanography*, pages 755–774. Elsevier, 2024.

[12] School of Information Technology and Electrical Engineering. Ml-based nids datasets. `https://staff.itee.uq.edu.au/marius/NIDS_datasets/`. Accessed: June 6, 2024.

[13] H. Benaddi, K. Ibrahimi, A. Benslimane, M. Jouhari, and J. Qadir. Robust enhancement of intrusion detection systems using deep reinforcement learning and stochastic game. *IEEE Transactions on Vehicular Technology*, 71(10):11089–11102, October 2022.

[14] J. Lansky et al. Deep learning-based intrusion detection systems: A systematic review. *IEEE Access*, 9:101574–101599, 2021.

[15] Z. Wang, D. Jiang, Z. Lv, and H. Song. A deep reinforcement learning based intrusion detection strategy for smart vehicular networks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2022.

[16] M. He, X. Wang, P. Wei, L. Yang, Y. Teng, and R. Lyu. Reinforcement learning meets network intrusion detection: A transferable and adaptable framework for anomaly behavior identification. *IEEE Transactions on Network and Service Management*, 21(2):2477–2492, April 2024.

[17] S. Dong, Y. Xia, and T. Peng. Network abnormal traffic detection model based on semi-supervised deep reinforcement learning. *IEEE Transactions on Network and Service Management*, 18(4):4197–4212, December 2021.

[18] T. Kim and W. Pak. Early detection of network intrusions using a gan-based one-class classifier. *IEEE Access*, 10:119357–119367, 2022.

[19] R. R. d. Santos, E. K. Viegas, A. O. Santin, and V. V. Cogo. Reinforcement learning for intrusion detection: More model longness and fewer updates. *IEEE Transactions on Network and Service Management*, 20(2):2040–2055, June 2023.

[20] X. Ma and W. Shi. Aesmote: Adversarial reinforcement learning with smote for anomaly detection. *IEEE Transactions on Network Science and Engineering*, 8(2):943–956, April–June 2021.

[21] M. Al-Fawa'reh, J. Abu-Khalaf, P. Szewczyk, and J. J. Kang. Malbot-drl: Malware botnet detection using deep reinforcement learning in iot networks. *IEEE Internet of Things Journal*, 11(6):9610–9629, March 2024.

[22] C. Picard and S. Pierre. Rlauth: A risk-based authentication system using reinforcement learning. *IEEE Access*, 11:61129–61143, 2023.

[23] Y. Feng, W. Zhang, S. Yin, H. Tang, Y. Xiang, and Y. Zhang. A collaborative stealthy ddos detection method based on reinforcement learning at the edge of internet of things. *IEEE Internet of Things Journal*, 10(20):17934–17948, October 2023.

[24] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, and M. Colajanni. Deep reinforcement adversarial learning against botnet evasion attacks. *IEEE Transactions on Network and Service Management*, 17(4):1975–1987, December 2020.

[25] S. Yu et al. Deep q-network-based open-set intrusion detection solution for industrial internet of things. *IEEE Internet of Things Journal*, 11(7):12536–12550, April 2024.

[26] S. Vadigi, K. Sethi, D. Mohanty, S. P. Das, and P. Bera. Federated reinforcement learning based intrusion detection system using dynamic attention mechanism. *Journal of Information Security and Applications*, 78:103608, November 2023.

[27] L. Chavali, A. Krishnan, P. Saxena, B. Mitra, and A. Sreevallabh Chivukula. Off-policy actor-critic deep reinforcement learning methods for alert prioritization in intrusion detection systems. *Computers & Security*, 142:103854, July 2024.

[28] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys & Tutorials*, 21(1):686–728, First quarter 2019.

[29] Y. Liu, M. Dong, K. Ota, J. Li, and J. Wu. Deep reinforcement learning based smart mitigation of ddos flooding in software-defined networks. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2018.

[30] C. Rookard and A. Khojandi. Rriot: Recurrent reinforcement learning for cyber threat detection on iot devices. *Computers & Security*, 140:103786, May 2024.

[31] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications*, 141:112963, March 2020.

[32] N. Mishra and S. Pandya. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, 9:59353–59377, 2021.

[33] M. Wang, N. Yang, N. J. Forcade-Perkins, and N. Weng. Progen: Projection-based adversarial attack generation against network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 19:5476–5491, 2024.

[34] C. Park, J. Lee, Y. Kim, J. G. Park, H. Kim, and D. Hong. An enhanced ai-based network intrusion detection system using generative adversarial networks. *IEEE Internet of Things Journal*, 10(3):2330–2345, February 2023.

[35] R. Soleymanzadeh and R. Kashef. Efficient intrusion detection using multi-player generative adversarial networks (gans): an ensemble-based deep learning architecture. *Neural Computing and Applications*, 35(17):12545–12563, March 2023.

[36] A. B. Mohammed, L. C. Fourati, and A. M. Fakhrudeen. Comprehensive systematic review of intelligent approaches in UAV-based intrusion detection, blockchain, and network security. *Computer Networks*, 239:110140, Feb. 2024.

[37] M. Samantaray, R. C. Barik, and A. K. Biswal. A comparative assessment of machine learning algorithms in the iot-based network intrusion detection systems. *Decision Analytics Journal*, 11:100478, June 2024.

[38] R. Tahri, Y. Balouki, A. Jarrar, and A. Lasbahani. Intrusion detection system using machine learning algorithms. In *ITM Web of Conferences*, volume 46, page 02003. EDP Sciences, 2022.

[39] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak. A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection. *IEEE Access*, 11:76071–76094, 2023.

[40] W. Zhong, N. Yu, and C. Ai. Applying big data based deep learning system to intrusion detection. *Big Data Mining and Analytics*, 3(3):181–195, September 2020.

[41] A. Sarıkaya, B. G. Kılıç, and M. Demirci. Raids: Robust autoencoder-based intrusion detection system model against adversarial attacks. *Computers & Security*, 135:103483, December 2023.

[42] L. Yang, J. Li, L. Yin, Z. Sun, Y. Zhao, and Z. Li. Real-time intrusion detection in wireless network: A deep learning-based intelligent mechanism. *IEEE Access*, 8:170128–170139, 2020.

[43] R. Ben Said, Z. Sabir, and I. Askerzade. Cnn-bilstm: A hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection. *IEEE Access*, 11:138732–138747, 2023.

[44] R. B. Said and I. Askerzade. Attention-based cnn-bilstm deep learning approach for network intrusion detection system in software defined networks. In *2023 5th International Conference on Problems of Cybernetics and Informatics (PCI)*, pages 1–5, 2023.

[45] M. H. Haghighat and J. Li. Intrusion detection system using voting-based neural network. *Tsinghua Science and Technology*, 26(4):484–495, August 2021.

[46] Y. Yang et al. Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in iiot environment. *IEEE Transactions on Network Science and Engineering*, 2022. Early access, March 8.

[47] L. Zou, X. Luo, Y. Zhang, X. Yang, and X. Wang. Hc-dttsvm: A network intrusion detection method based on decision tree twin support vector machine and hierarchical clustering. *IEEE Access*, 11:21404–21416, 2023.

[48] B. Sharma, L. Sharma, C. Lal, and S. Roy. Anomaly based network intrusion detection for iot attacks using deep learning technique. *Computers and Electrical Engineering*, 107:108626, April 2023.

[49] A. Sharma and H. Babbar. Luflow: Attack detection in the internet of things using machine learning approaches. In *2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pages 1–5, 2023.

[50] X. Lou, P. Li, N. Sun, and G. Han. Botnet intrusion detection method based on federated reinforcement learning. In *2023 International Conference on Intelligent Communication and Networking (ICN)*, pages 180–184, 2023.

[51] N. Niknami and J. Wu. Deepidps: An adaptive drl-based intrusion detection and prevention system for sdn. In *ICC 2024 - IEEE International Conference on Communications*, pages 2040–2046, 2024.

[52] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut. A hybrid cnn-lstm based approach for anomaly detection systems in sdns. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*. ACM, August 17 2021.

[53] K. Jiang, W. Wang, A. Wang, and H. Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access*, 8:32464–32476, 2020.