Deep Reinforcement Learning for Intrusion Detection and Prevention System in Internet of Things Devices

Erfan Nosrati[1], Ehssan Mousavipour[1] and Adrian Tadicć[1]

[1] *Department of Computer Science, University of Calgary*
2500 University Dr NW, Calgary, AB T2N 1N4

E–mail: *erfan.nosrati@ucalgary.ca, ehssan.mousavipour@ucalgary.ca, adrian.tadic@ucalgary.ca*

Abstract

The increase of Internet of Things (IoT) devices across diverse sectors has brought convenience and efficiency, alongside substantial security challenges. These devices, limited by computational resources, are vulnerable to cyber-attacks, underscoring the urgent need for effective Intrusion Detection and Prevention Systems (IDPS) tailored to the IoT context. Traditional IDPS solutions, however, are often infeasible for direct deployment on IoT devices due to these computational constraints. In response, we introduce a novel Intrusion Detection and Prevention System (IDPS) that harnesses the power of Machine Learning (ML) and Deep Reinforcement Learning (DRL) to secure IoT networks against a broad spectrum of cyber threats. Our IDPS is designed to function on network servers and aggregate logs from IoT devices to detect and neutralize attacks, thereby overcoming the devices' processing limitations and offering a scalable, adaptable defense mechanism. The system incorporates a four-phase ML pipeline, beginning with Random Forest algorithms for device classification, followed by a Convolutional Neural Network (CNN) for attack detection, an XGBoost model for attack type identification, and finally, an autoencoder-autodecoder model to test the system robustness by generating new data. This paper presents a dual contribution: firstly, establishing an IDPS framework that integrates sophisticated ML methodologies to secure IoT environments effectively; and secondly, executing a proof-of-concept Reinforcement Learning (RL) model that autonomously responds to detected threats by issuing corrective directives to affected IoT devices, thereby demonstrating a proactive approach to IoT security.

Key words. Internet of Things – Intrusion Detection – Intrusion Prevention – Network Security

1. Introduction

The growth of Internet of Things (IoT) devices in various sectors, from industrial applications to domestic environments, has started a new era of convenience and efficiency. However, this widespread adoption of IoT technologies also presents significant security challenges. IoT devices, often constrained by their computational capabilities, become prime targets for cyber-attacks, compromising user privacy, data integrity, and service availability. Lack of security emphasizes the critical need for robust Intrusion Detection and Prevention Systems (IDPS) designed for the IoT ecosystem. Yet, the limited processing power of many IoT devices restricts the deployment of traditional IDS solutions directly on these devices. Addressing these concerns, we propose a novel Intrusion Detection and Prevention System (IDPS) that leverages Machine Learning (ML) and Deep Reinforcement Learning (DRL) to protect IoT networks from a wide array of cyber threats.

Our IDPS is built to function on network servers that collect log information from IoT devices. It serves two main roles: identifying attacks aimed at IoT devices and setting up measures to counteract the attack. This strategy not only overcomes the processing restrictions of IoT devices but also offers a scalable, adaptable way to protect IoT ecosystems from emerging cyber threats.

Furthermore, we enhance the IDPS with a four-phase ML pipeline, starting with the use of Random Forest algorithms to classify IoT devices based on their features. This initial classification lays the groundwork for subsequent phases, where a Convolu-

tional Neural Network (CNN) is employed to determine whether given data represents an attack. Following this, an XGBoost model is utilized to identify the specific type of attack, providing detailed insights into the threat landscape faced by the IoT network. The pipeline culminates in the training of an autoencoder model, which generates new data for testing the robustness of our system. This mainly serves as a simulator, since testing our solution in real-time is challenging and out of the scope of this course.

Our contributions are twofold: first, we introduce an IDS framework that integrates advanced ML techniques to safeguard IoT environments effectively. Second, we implement a proof of concept Reinforcement Learning (RL) model to respond to attacks by issuing correction orders to the IoT devices.

2. Motivation

The Internet of Things (IoT) is rapidly transforming our environment, enabling smarter ecosystems in urban planning and various industrial domains. This transformation is powered by integrating countless devices, from the simplest sensors to complex processing units, into a globally interconnected network. Such connectivity, while offering unprecedented opportunities for automation and efficiency, also introduces significant security vulnerabilities. (6) The lightweight hardware of IoT devices makes them vulnerable targets for cyber threats, with potential consequences ranging from data breaches to the disruption of critical infrastructure services.

Recent cyber incidents, including the infamous Mirai (7) botnet attack, have highlighted the vulnerability of existing IoT infrastructures under orchestrated cyber-attacks. These incidents have compromised the privacy and security of individual users and demonstrated the potential for widespread disruption affecting major online services. The challenge, however, lies in addressing the unique characteristics of IoT ecosystems, which include a vast array of device capabilities, the limited computational resources of many IoT components, and the complex network of interactions between devices and users.

Our research responds to this challenge by proposing an advanced Intrusion Detection and Prevention System (IDPS) designed for the IoT ecosystem. Leveraging the power of machine learning and deep reinforcement learning, our system is designed to dynamically classify devices, detect potential attacks, identify the nature of these threats, and generate synthetic data to continuously enhance its detection capabilities. This multi-faceted approach aims to fortify IoT networks against a wide spectrum of cyber threats including DDoS and MITM, ensuring the integrity, confidentiality, and availability of services.

3. Background

In this section, we are going to discuss the necessary background for readers to understand our work. We will first discuss the Random Forest (RF) algorithm, The Convolutional Neural Network (CNN), XGBoost, Autoencoders, and finally the details of our Deep Deterministic Policy Gradient (DDPG) which is a form of Reinforcement Learning (RL) algorithm.

3.1. Random Forest

Random Forest (RF) is an ensemble learning method for classification, regression, and other tasks, which operates by constructing a multitude of decision trees at training time and outputting the mode of the classes (for classification) or mean prediction (for regression) of the individual trees. This method combines the simplicity of decision trees with flexibility, resulting in a robust model that can handle large datasets with a high dimensionality of features. RF corrects for decision trees' habit of over-fitting to their training set, making them particularly well-suited for identifying the types of devices in an IoT network, where the variability and complexity of data require both accuracy and Adaptability ((8)).

3.2. Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a class of deep neural networks, most commonly applied to analyzing visual imagery, that have been successfully adapted for a variety of tasks beyond image processing, including time-series analysis and natural language processing. Characterized by their unique architecture of convolutional layers, pooling layers, and fully connected layers, CNNs are adept at automatically and adaptively learning spatial hierarchies of features from input data. The convolutional layers act as feature extractors, while the fully connected layers interpret these features for classification or regression tasks. In the context of IoT security, after the device type is determined using RF, a CNN can further analyze the device-specific features to effectively detect whether the devices are under attack. This two-stage approach leverages the strengths of both models: the feature engineering and interpretability of RF and the high-dimensional pattern recognition capability of CNNs, offering a comprehensive solution for securing IoT networks ((11)).

3.3. XGBoost

XGBoost, an acronym for Extreme Gradient Boosting, is a highly efficient and scalable implementation of gradient-boosted trees designed for speed and performance. It is a popular machine learning algorithm that has gained widespread recognition for its capability to tackle large-scale and complex data challenges. XGBoost works by constructing an ensem-

ble of decision trees sequentially, where each successive tree corrects errors made by the previous ones, thereby improving the model's accuracy. Its versatility allows it to be applied across a wide range of regression, classification, and ranking problems, making it a favored choice among researchers and practitioners in various domains, including the detection of types of attacks on IoT datasets. If the CNN determines a device is under attack, we will pass this device along with its features to XGBoost to determine what type of attack we are dealing with ((2)).

3.4. AutoEncoder (AE)

Autoencoders (AE) are a type of generative artificial intelligence that comprises an Encoder and a Decoder. Initially, an input is fed into the Encoder, which then produces a hidden layer; this serves as a compressed, lower-fidelity representation of the input. Subsequently, this hidden state h is passed to the Decoder, with the aim for the Decoder to reconstruct an output that is identical to the input. This is achieved by minimizing the Mean Squared Error (MSE) between the input x and the Decoder's output $\hat{x}$, optimizing the process until the discrepancy between x and $\hat{x}$ becomes negligible. Upon successfully training both the Encoder and Decoder, the Decoder is then utilized to process Gaussian noise, enabling it to generate instances that mimic the distribution of our IoT data. This functionality allows us to evaluate the robustness of our model, acting as an IoT simulator. It ensures that our Intrusion Detection and Prevention System (IDPS) maintains high performance on unseen data, which is synthesized by the Autoencoder ((3)).

3.5. Reinforcement Learning (RL)

Since Reinforcement Learning (RL) is the most important part of our work, we will delve deep into its underlying algorithm. RL is a class of machine learning where an agent learns to make decisions by receiving feedback from its environment. This section is entirely the contribution of (5). These are 5 main components in all RL algorithms:

- Agent: The entity that learns and makes decisions in the RL framework (in Deep RL this is the neural network that makes the decisions).
- Action: The set of all possible moves or decisions an agent can make.
- Environment: The world through which the agent moves and interacts.
- State: The current situation or context within the environment that the agent perceives.
- Reward Function: A rule or formula to evaluate the success of an action, guiding the agent's learning.

The simplest version of RL which does not include neural networks is called Q-learning and is a form of Dynamic Programming. In more complex algorithms of RL which are considered Deep RL (DRL), we use neural networks to estimate how good or bad the agent is doing. The specific algorithm we are going to use is called Deep Deterministic Policy Gradient (DDPG). The reason why we chose this algorithm is that it is sample efficient, which means it can achieve its goal with fewer samples, and also the actions it chooses are deterministic which is exactly what we need for our work. Below, we delve deep into what this algorithm does and how it works.

3.5.1. Deep Deterministic Policy Gradient (DDPG)

Before we can dive into mathematical formulas, we need to define a couple of terms in order for the reader to understand how this algorithm works.

- Q-value or Q-function $Q_\theta(s, a)$: This is a function parameterized by θ (it is a neural network) that gives us the value of taking action a in state s. In other words, it tells us how good or bad it is for us to take action a in the state s. We also call this Critic Network as it criticizes the actions of our agents.
- Policy or Policy network ($\mu_\phi(a)$): This is a function parameterized by ϕ that learns the policy on which the agent takes the action based on. In other words, it is a neural network that defines the agent's strategy and maps states to actions. We also call this network the Actor Network as it takes actions in the environment
- Target Networks: These are the replicas of Critic and Actor Networks. The reason behind having these two networks is that we will train them so that the Target Critic Network tells us how good the current action and state are for us in the long term, and the Target Actor Network can be able to take action in future states. We will explain the logic and intuition behind them later on.
- Replay Buffer: This is a memory that takes stores $State_{Current}$, $Action_{Current}$, $Reward$, $State_{Next}$. These samples are going to be used to train the Target Critic Network. In other words, by training the Target Critic Network, it will be able to tell us how good it is to take a specific action in a specific state in the long term.

Now that we have these definitions, we will go through the math and optimization part of this algorithm.

As we mentioned, the agent takes an action and receives a reward based on how good or bad it performed its actions. For that, we need to define a formula to compute the Q-value. The formula is defined as:

$$Q_\theta(s, a) = r + \gamma(Q_{\theta_{target}}(s', a')) \tag{1}$$

Where is the value estimate of taking action in state , r is the current reward we received from taking action in state , is a discount factor, and is the future value estimate. In simpler terms, we want to make sure that by taking action in the current state, we get the most possible reward from now onwards. The discount factor here says that even though we care about future rewards associated with our current states and actions, we want to discount them regarding the current reward we can get, which is r.

There is only one glitch here. The left side of the equation is not equal to the left part at first. But our final goal is to make them equal because we want to make sure if we are taking action a in the state s, it is going to yield the most value in the long term. Therefore, we have to solve the optimization problem below:

$$\nabla_\theta \frac{1}{|B|} \sum_{(s,a,r,s') \in B} (Q_\theta(s,a) - (r + (Q_{\theta_{target}}(s',a'))))^2 \tag{2}$$

Where B is the Replay Buffer. In simpler terms, we compute the gradient of the loss function, which measures the discrepancy between our current estimate of the Q-value $Q_\theta(s,a)$ and the target Q-value, comprised of the immediate reward r plus the discounted future Q-value $Q_{\theta_{target}}(s',a')$ across all transitions (s,a,r,s') sampled from the Replay Buffer. This gradient is instrumental in adjusting θ to minimize the error between the predicted and target Q-values, thereby refining the policy network's accuracy and the overall decision-making capability of the agent. It's noted that in the formula for the future Q-value, we encounter a', which represents the action to be taken in the future state, yet its origin has not been explicitly defined. This action, a', is determined by the Target Actor Network, whose specific role is to select the optimal future action based on the given future state. This mechanism underscores the collaborative function of the Target Actor Network in our model, ensuring that actions are chosen with foresight and aligned with the evolving policy for improved decision-making.

Now that we discussed the optimization of the Q-value function, we can go through the policy function. As we discussed earlier, we want the policy function to choose the action that yields the most reward. In other words, we are trying to solve the below optimization problem for the policy network:

$$\max_\phi \mathop{\mathbb{E}}_{s \sim D} [Q_\theta(s, \mu_\phi(s))] \tag{3}$$

Where s represents a state sampled from the environment, Q_θ denotes the Q-value function, and $\mu_\phi(s)$, parameterized by ϕ, refers to the policy network designed to select the optimal action for a given state. Our objective is to enhance the Q-value by optimizing this policy network. Essentially, we aim to increase the expected outcome of the Q-value function, training the policy network to recommend the most advantageous action when presented with a specific state.

Identifying the optimal parameters for the policy network to maximize the expected Q-value presents a significant challenge. This optimization problem can be addressed by employing gradient ascent on the Q-value function. Contrary to gradient descent, which minimizes a function by moving in the direction of the steepest decrease, gradient ascent involves updating the parameters ϕ of the policy network in the direction of the steepest increase of the Q-value. Specifically, if gradient descent aims to minimize the Q-value by updating ϕ in the direction of the negative gradient, to maximize the Q-value, we apply the principle of gradient ascent, effectively adjusting ϕ in the direction of the positive gradient of the Q-value function. This approach systematically guides the optimization process toward maximizing the expected Q-value, thereby resolving the optimization challenge. As a result, this is a new optimization problem which is significantly easier to compute:

$$\nabla_\phi \frac{1}{|B|} \sum_{s \in B} (Q_\phi(s, \mu_\phi(s))) \tag{4}$$

As noted, solving the above optimization problem is easier, and we just have to take the negative of the above formula and use it as the loss function to update the parameters of the policy network to maximize the Q-value.

With this conclusion, we can go through previous works done in the field and how our methodology works.

4. Related Works

4.1. Intrusion Detection using Machine Learning

The paper by Tanzila Saba(11). introduces a significant advancement in IoT security through a deep learning-based anomaly detection system. Highlighting the necessity for improved security in the increasingly interconnected IoT environment, the proposed CNN model efficiently detects potential intrusions by analyzing network traffic. Evaluated using the NID and BoT-IoT datasets, the model achieved accuracies of 92.85% using the BoT-IoT Dataset, showcasing the effectiveness of deep learning techniques specifically CNN models in addressing IoT security challenges. Building upon this foundational work, our study employed the same datasets. It achieved better results in identifying abnormal traffic, marking a significant advancement in developing adaptive and robust security solutions for IoT networks.

The paper by Eirini Anthi (1). extends the realm of IoT security further by introducing an innovative three-layer Intrusion Detection System (IDS), meticulously designed to fortify the security framework of

smart home IoT devices. This system seamlessly integrates supervised machine learning techniques to meticulously categorize and profile the regular behavior of IoT devices, proficiently identify malicious data packets, and accurately classify the type of cyber-attacks. Moreover, this IDS excels in its capacity to differentiate between distinct IoT devices and efficiently detect network-based cyber threats, classifying them into specific categories such as Denial of Service (DoS), Man-in-the-Middle (MITM), and spoofing. Their system archives success, underlining the potent capability of machine learning in mitigating the complex challenges associated with IoT security.

4.2. Intrusion Prevention using Reinforcement Learning

Our project utilizes Reinforcement Learning (RL) within the domain of IoT security, specifically in the prevention of cyber-attacks. By leveraging RL models, our system not only detects but also proactively mitigates attacks by dynamically issuing corrective rules to IoT devices. This adaptive approach allows for the continuous learning and improvement of the system's defense mechanisms based on feedback from its interactions with the environment, thereby enhancing the resilience of IoT networks against threats. To the best of our knowledge, this innovative application of RL for direct attack prevention and rule-based mitigation in IoT devices represents a novel contribution to the field. It underscores our project's commitment to advancing the frontier of IoT security solutions, bridging a critical gap in existing research that has primarily focused on attack detection and classification, rather than prevention.

4.3. Data Generation and Attack Type Identification in IoT Security

In conclusion, our review of related work highlights the advancement in IoT security strategies, from deep learning for anomaly detection to complex intrusion detection systems utilizing machine learning for detailed network traffic analysis and attack identification. Our project contributes to this field by integrating reinforcement learning (RL) for proactive attack prevention and using XGBoost algorithms for accurate attack classification. Additionally, we've incorporated auto-encoders to generate new data, enhancing our model's robustness. This multifaceted approach sets our work apart from traditional methods focused mainly on detection, marking a significant move towards developing adaptive, resilient security solutions that not only anticipate but also precisely identify cyber threats. This innovation opens new paths for research, aiming to improve IoT security against increasingly sophisticated threats and suggests a shift towards proactive, precise security frameworks, potentially redefining IoT security standards.

5. Methodology

For training our model, we employed the BoT-IoT Dataset (7), using the four-layer pipeline methodology as suggested by (1). The specifics of each layer in this methodology will be outlined in the sections that follow. Initially, the first layer processes data from various IoT devices, preparing it for further analysis. In the next layer, a classifier model is deployed to identify the type of device from the pre-processed data. In the third layer, we classify the data created from the previous layer to decide if it is normal or indicates a potential attack. If the data is classified as an attack, it is then forwarded to another model to identify the specific type of attack encountered. If we encounter any attacks, we employ our Reinforcement Learning model to respond appropriately to mitigate the damage caused by the attack and prevent it.

5.1. Intrusion Detection System

5.1.1. Preprocessing

The BoT-IoT dataset includes information gathered from seven different IoT devices: Fridge, Garage, GPS Tracker, Modbus, Motion Light, Thermostat, and Weather Sensors. It also includes a status for each data entry indicating whether it represents an attack, and if so, the type of attack. In the preprocessing stage, we merge data from these devices. For device-specific parameters that are undefined for others, we assign an out-of-range value to maintain consistency across the dataset.

To facilitate the analysis and enable our models to interpret the categorical data effectively, we apply one-hot encoding to the dataset. Then, to optimize the features within a specific range, we utilize standardization. This process scales our data to a range with a standard deviation of 1 and a mean of 0, enhancing the feature optimization. For training and testing our model effectively, we need to split the dataset into two parts. One portion, comprising 80% of the data, is allocated for training, while the remaining 20% is reserved for testing.

5.1.2. Device Identifier

In our study, we address the challenge of accurately identifying the type of device from data sourced in IoT applications. To achieve this, we deploy the Random Forest algorithm which is efficient in both classification and regression tasks. This ensemble learning technique mitigates the risk of over-fitting, a common issue in predictive modeling, by synthesizing the predictions from numerous decision trees. A significant advantage of employing Random Forest in our context is its resilience to datasets with missing or null values. This characteristic is particularly beneficial

for processing real-world IoT data, which often comes with its share of inconsistencies and gaps.

5.1.3. Attack Detector

To identify attacks within our network, we utilize a Convolutional Neural Network (CNN) model. This decision is influenced by the notable success of the CNN-IDS model in (4), where the model exhibited exemplary performance on the CIC-IDS2017 dataset which is another IoT database. By adopting the CNN model to our specific needs, we aim to leverage its capabilities for accurate attack identification.

The CNN architecture is instantiated with a convolutional layer designed to process one-dimensional input data, which is essential for analyzing sequential network traffic data effectively. This layer employs 64 filters to capture spatial features across the data with a kernel size of 3, maintaining the dimensionality through padding. The network further includes fully connected layers to filter these features into a more abstract representation, crucial for the classification task.

Our model utilizes dropout layers with varying probabilities to prevent over-fitting, a common challenge in deep learning models. This approach randomly deactivates a subset of neurons during training, thereby enhancing the model's generalization capability to unseen data. The final output is processed through a sigmoid activation function, providing a probability score indicative of the presence of a network intrusion.

5.1.4. Attack Identifier

Previously, we have determined if a specific device is being attacked or not. Then, the result of the previous step is passed to the XGBoost model to determine what the type of attack is. This is very crucial as the prevention mechanism chosen by the RL heavily relies on the type of attack. The reason for choosing XGBoost for detecting attack type is its power in classification tasks. Readers can refer to the evaluation section to examine the performance of the proposed XGBoost.

5.2. Intrusion Prevention

The last part of our contribution is the prevention mechanism and algorithm we use to take action when an attack is initiated on the devices. It is important to note that when we talk about prevention, we refer to taking a set of actions to prevent other devices from being affected or damaging to the network infrastructures. For this course, we are only considering DDoS and Ransomware attacks as taking all different categories of attacks is out of the scope of this course and takes a significant effort. We are creating an offline simulator to simulate a network of IoT devices using the Autoencoder that generates data for different device types. Furthermore, the focus of this paper is mainly on the log files of these devices rather than flow traffic. However, our proposed solution can be adapted and used for network flow traffic as well.

5.2.1. The Agent

As we discussed earlier, DDPG has an agent that makes the decisions. Below is how we define RL components for our work:

- Environment: The environment is the offline simulator that generates IoT logs based on their type
- States: The states are the features of these devices and they will change when they are being attacked
- Agent: The agent is a Deep Neural Network that takes appropriate actions based on the states of the devices
- Reward: The reward function is based on how well the agent was able to take actions based on types of attacks and degree of disruption
- Action: The action is the mitigation strategies chosen by the agent

Now that we have defined all the important components of our DDPG algorithm, we can delve into the details of our algorithm. First, the simulator will generate the logs for chosen devices–the details of what devices we used will be discussed in the evaluation section. Second, we will extract these features, and pass them to our intrusion detection pipeline. Suppose the logs indicate that the specified device is under a DDoS attack. In that case, the agent should adjust the filtering criteria to identify and drop packets that are likely part of the DDoS attack. This will help to mitigate the attack with minimal damage. The action space is between -1 to 1 and the way we interpret this is by taking value 0 to 1 as mitigation strategy for DDoS attack, and -1 to 0 as mitigation strategy for Ransomware attack. The values indicate how aggressively we want to apply our mitigation strategy. For example, for DDoS, if the agents output a value close to 1 it aims to apply more aggressive traffic filtering and vice versa.

The important part of this solution is the reward function. We have chosen the reward functions for penalizing wrong actions and rewarding the right actions. If the action was chosen correctly based on the type of the attack, we would give a reward of +10; If the false positive rate (falsely categorizing normal traffic as an attack) is beyond the acceptable threshold, we give a reward of -5. For the acceptable threshold, we chose the result F1 score of our XGBoost algorithm and the threshold will be $100 - F1Score$. Finally, we will penalize the agent with the below formula:

$$reward = reward - (degree\_of\_disruption \times 10) \quad (5)$$

Where degree_of_disruption measures how much the defensive actions taken by the system (e.g., the actions decided by the RL agent) disrupt normal system operations or legitimate user activities. This is crucial as it penalizes the agent heavily if its actions are too aggressive. As computing degree_of_disruption is not our contribution, we refer the readers to the paper ((9)) for further details.

After enough training, our agent can take the appropriate actions based on the type of attacks, and again we use the simulator to generate new logs to test the performance of our model. The details of the training process are in the evaluation section.

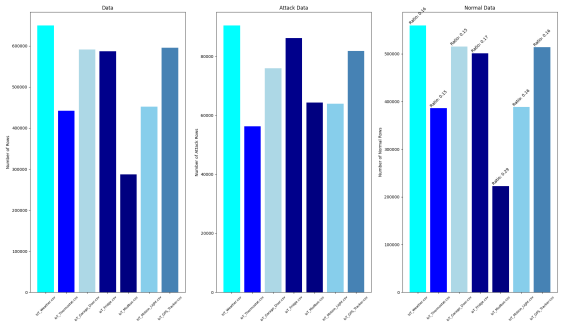

Fig. 1: The dataset's data distribution reveals a ratio of attack to normal data at around 0.15, indicating that the data is not evenly distributed across categories. Consequently, we are dealing with an imbalanced dataset in terms of both attacks and categories.

6. Evaluation

This section presents the evaluation results, demonstrating the effectiveness of our proposed solution and verifying the accomplishment of our objectives. Initially, we discuss the outcomes of our Intrusion Detection System (IDS), highlighting its success in detecting attacks and classifying their types. Subsequently, we explore our prevention model, assessing its performance based on the actions it takes to mitigate attacks. Due to constraints beyond the scope and time frame of this project, our evaluation of the prevention system is limited to two types of attacks: DDoS and Ransomware. All experiments were conducted on a personal computer equipped with a Core-i7 processor and a 3080 Ti GPU, featuring 12GB of video RAM.

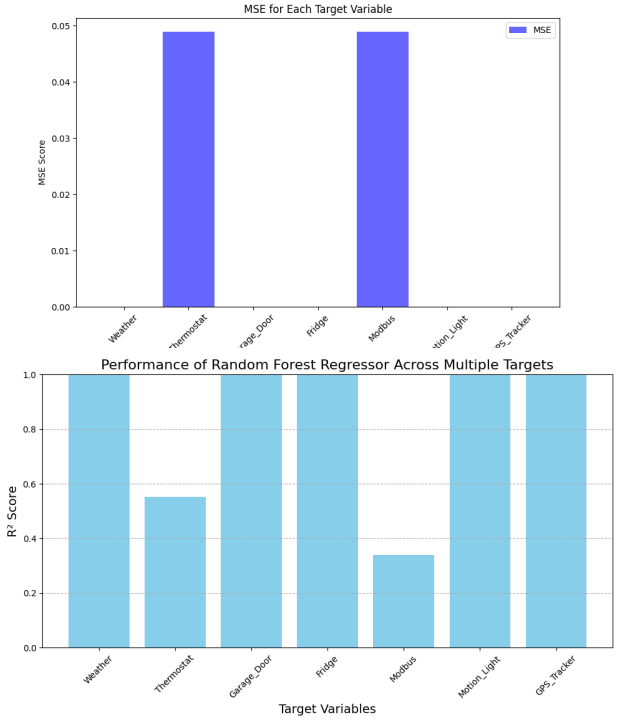

Fig. 2: Random-forest MSE and R^2 values.

6.0.1. Device Identifier

As discussed earlier, we utilize the BoT-IoT dataset, and 1 illustrates that this dataset is imbalanced, with the ratio of attack to normal being around 0.15, which can lead to model bias or poor performance in attack classifications. However, the RandomForest model successfully classifies the device data 2. We are using R^2 values to assess how well the variation of the dependent variable is explained by the model and its interaction with other types of devices (10) and the MSE to prediction loss by averaging the squares of the errors between actual and predicted values. This approach helps to determine the effectiveness of our model. The MSE value is near zero for all devices, and theR^2 value is 1 for all except the thermostat and Modbus, which is also acceptable due to the imbalanced dataset properties.

6.0.2. Attack Detector

we used Recall, Precision, F1-score, and Accuracy to evaluate the performance of our proposed solution, Figure 3. Additionally, we utilized a confusion matrix to visualize our model's performance. In Figure 4, the confusion matrix depicts the classification performance of our model, specifically in distinguishing between benign and attack traffic. The model

has demonstrated commendable precision in identifying actual attack instances, as evidenced by the high number of true positives (6e+06), crucial for robust cybersecurity measures. Conversely, the true negative count (1e+05), albeit lower due to the dataset attack data, indicates a reasonable ability of the model to recognize benign traffic. Moreover, zero false positives and false negatives demonstrate that the model accurately classifies the attack and benign traffic.

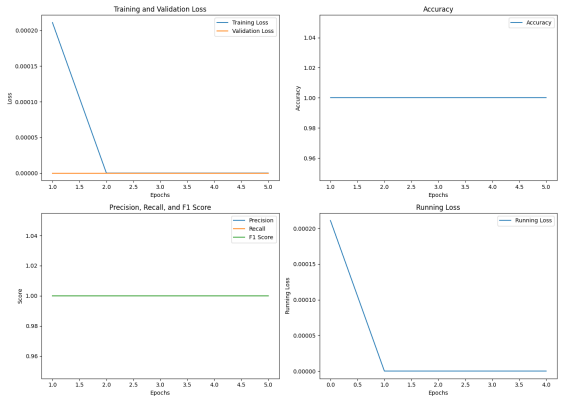

Fig. 3: The CNN can classify the attack data from the normal data after the first epoch.

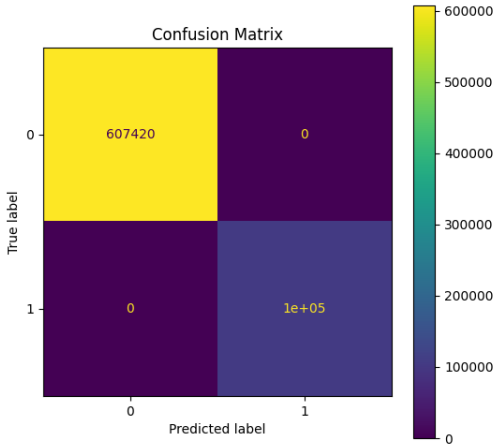

Fig. 4: The confusion matrix is a powerful tool that compares the actual target values with those predicted by the model, allowing for a detailed analysis of accuracy.

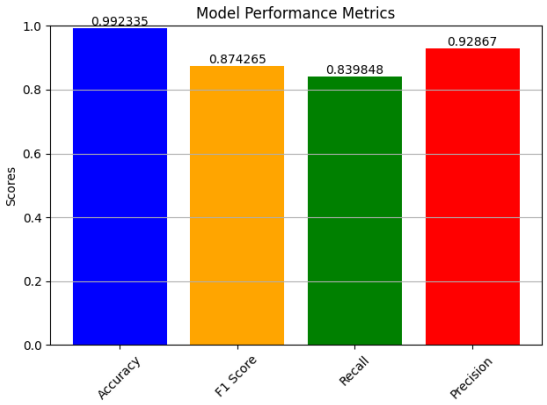

Fig. 5: ?????XGBoost metrics show that it can identify the type of attack.

6.0.3. Attack identifier

Utilizing the same metrics as outlined in the previous section, our XGBoost model exhibits exceptional capability in accurately identifying the type of attack, a testament to its efficiency. This is clearly demonstrated in Figure 5, underscoring the successful classification performance of our proposed method. Figure 6 demonstrates the model loss metrics during training and evaluation phases across epochs. We observe a sharp decrease in loss, suggesting rapid learning and improvement in the initial epochs in both training and validation, which then plateaus, demonstrating the model's ability to generalize as the loss decreases with additional epochs, also illustrating the effectiveness of the training process.

6.0.4. intrusion prevention

As mentioned earlier, we employ autoencoder models to generate data that enhance the robustness of our Intrusion Detection System (IDS) before its application in our Reinforcement Learning (RL) model. Figure 7 shows the generated data (in red) alongside the original data (in blue), demonstrating that the model successfully creates data that is similar, but not identical, to the training set distribution. To reduce the dimensionality of the features to plot them, we apply Principal Component Analysis (PCA), a statistical technique that simplifies data by reducing its dimensions while retaining most of the original variability, here showing the first two principal components of these data sets. PCA helps us to see how close two different distributions are (the generated and original data).

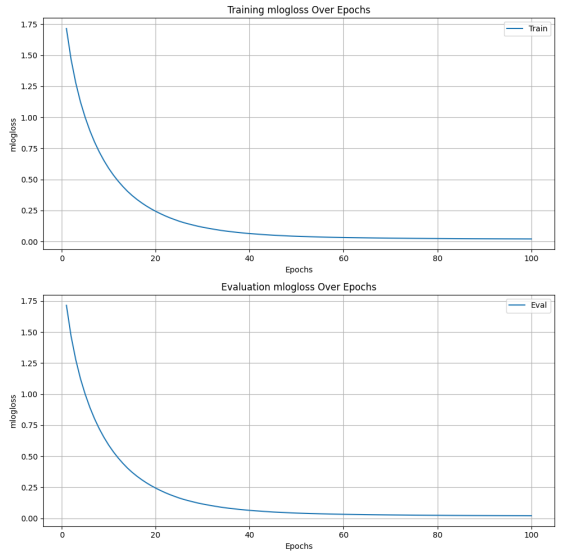

Fig. 6: The CNN can classify the attack data from the normal data after the first epoch.

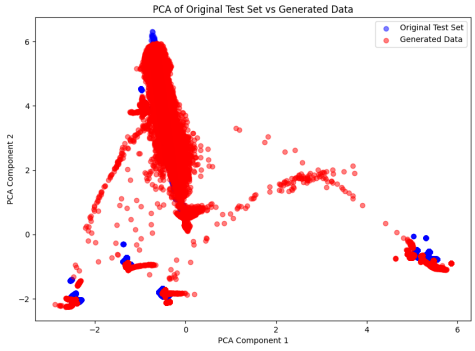

Fig. 7: The CNN can classify the attack data from the normal data after the first epoch.

We use our model metrics as well as our model output in this section to evaluate our RL model. The first plot, actor loss, shows a trend of increasing loss values over training iterations, suggesting the actor is continuously exploring and adjusting its policy. As we mentioned previously, the actor loss is the negative of the gradient of the Q-value function, therefore, it is normal for the loss to increase, we are taking the gradient ascent, not the gradient descent for actor loss. The second plot shows the critic loss, which decreases sharply initially and then stabilizes, showing that the critic network is effectively learning to evaluate the policy's actions and the target network. Finally, the reward per episode plot demonstrates an upward trend in cumulative rewards, signifying that the model's policy is initially negative, allowing the agent to explore possible actions, and then improve, successfully maximizing rewards over time.

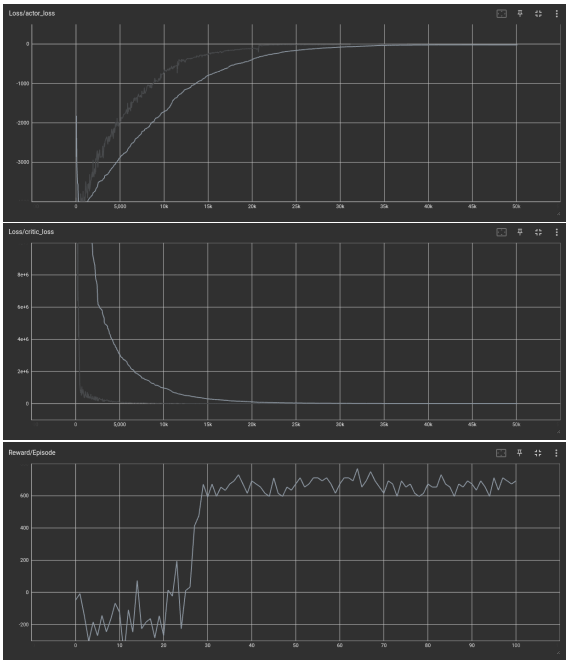

Fig. 8: From top to bottom: Actor loss, Critic loss, Rewards.

1 shows that RL was able to mitigate DDoS attacks initiated on Fridge and Thermostat. However, the RL showed a poor performance in mitigating Ransomware attacks initiated on Thermostat sensors. This is mostly because the AE generated more data for Fridge and among those instances that were anomalies, most of the logs demonstrated instances of DDoS. As a result, RL was exposed to more instances of DDoS, which explains why it could not mitigate Ransomware.

To illustrate how we achieved these results, let's examine the operation of the prevention component of our pipeline. Initially, Autoencoders (AE) are utilized to generate logs. These logs are then fed into a Random Forest (RF) model, which identifies the type of device involved, such as a Fridge. Subsequently, the output from RF is processed by a Convolutional Neural Network (CNN), which assesses a high probability that the device is under attack. The findings from the CNN are further analyzed by an XGBoost model, determining the presence of a DDoS attack. This information is finally passed to a Reinforcement Learning (RL) system, which suggests "Adjusting the filtering criteria to identify and drop packets likely part of the DDoS attack." as a mitigation strategy. This demonstrates our model's capability to discern appropriate countermeasures. However, to validate its effectiveness, testing in a real IoT device network

DEVICE TYPE	ATTACK DETECTED	ACTION	Result
Fridge	type_ddos	Packet Drops	ATTACK MITIGATED
Thermostat	type_ransomware	reducing the rate of requests	ATTACK NOT MITIGATED
Thermostat	type_ddos	Packet Drops	ATTACK MITIGATED

Table 1: Summary of Attacks and Mitigation

and directly applying the RL output in the environment is necessary.

7. Conclusion

In this paper, we delve into the application of Deep Reinforcement Learning (DRL) for enhancing Intrusion Detection and Prevention Systems (IDPS) on the Internet of Things (IoT), highlighting its capacity to adapt and respond to the evolving cyber threat landscape. Through our comprehensive review, we have identified that DRL's dynamic learning capabilities offer significant advancements over traditional machine learning methods, especially in handling the IoT's complexity and variability.

Additionally, we have improved the performance of the IDS model by using the XGBoost model to identify attacks with great precision. Key insights emphasize the importance of balancing detection accuracy with computational efficiency and the ongoing challenges related to data privacy and real-time processing needs.

In conclusion, DRL presents a promising pathway for securing IoT environments against cyber threats. Future research focusing on the identified challenges will be crucial in realizing the full potential of DRL-based IDS in IoT security.

8. Future Works

Our study focused on using log files for our learning models applied to intrusion detection and prevention systems (IDPS) in IoT, setting the stage for further exploration. Future efforts could expand on this by incorporating network flow data, potentially unveiling more cyber threats and enhancing model performance. Additionally, while our research utilized a simulator, subsequent studies should aim to implement and evaluate DRL-based IDPS models in real IoT environments. This shift from simulation to practical application will allow for a broader assessment across various DRL models and attacks and enhance the prevention of attacks other than DDoS, offering deeper insights into their effectiveness in real-world scenarios.

Exploring the optimization of DRL algorithms for IoT's constrained resources and investigating federated learning for privacy concerns are also critical future directions. Moving forward, advancing DRL-based IDS, to fully meet IoT security needs, will require innovative approaches addressing these identified areas.

REFERENCES

[1] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019.

[2] Mauro AA da Cruz, Lucas R Abbade, Pascal Lorenz, Samuel B Mafra, and Joel JPC Rodrigues. Detecting compromised iot devices through xgboost. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[3] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. Lifelong anomaly detection through unlearning. pages 1283–1297, 2019.

[4] Asmaa H. Halbouni, Teddy Surya Gunawan, Murad Halbouni, Faisal Ahmed Abdullah Assaig, Mufid Ridlo Effendi, and Nanang Ismail. Cnn-ids: Convolutional neural network for network intrusion detection system. pages 1–4, 2022.

[5] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. 2019.

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

[7] Nour Moustafa. The bot-iot dataset, 2019.

[8] Paulo Angelo Alves Resende and André Costa Drummond. A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys (CSUR)*, 51(3):1–36, 2018.

[9] Maryam Roshanaei. Resilience at the core: critical infrastructure protection challenges, priorities and cybersecurity assessment strategies. *Journal of Computer and Communications*, 9(8):80–102, 2021.

[10] Souradip Roy, Juan Li, Bong-Jin Choi, and Yan Bai. A lightweight supervised intrusion detection mechanism for iot networks. *Future Generation Computer Systems*, 127:276–285, 2022.

[11] Tanzila Saba, Amjad Rehman, Tariq Sadad, Hoshang Kolivand, and Saeed Ali Bahaj. Anomaly-based intrusion detection system for iot networks through deep learning model. *Computers and Electrical Engineering*, 99:107810, 2022.