Enhancing Cybersecurity in the Internet of Things through Machine- and Deep Learning: A Novel Approach to Threat Detection

JELKE SCHRÖDER, University of Twente, The Netherlands

The Internet of Things (IoT) is growing rapidly each day. By the increase of the number of internet-connected devices, security threats are getting more severe as well. Different machine learning models have been trained on malicious network traffic to detect security risks. In this study, we will test five machine learning techniques by manipulating training data to simulate a novel malware attack. Furthermore, we evaluate whether the trained models retain the ability to detect instances of excluded and therefore unobserved attacks in the training data. The Recurrent Neural Network performed best, with a weighted accuracy of 0.8173, followed by a Multilayer Perceptron Network with a weighted accuracy of 0.7828. This enhances our understanding of which machine learning techniques are most capable of detecting currently unknown types of attack and will improve our capabilities for future security.

CCS Concepts: • **Computing methodologies** → **Machine learning approaches**; *Machine learning algorithms*;

Additional Key Words and Phrases: IoT, anomaly detection, unknown attacks, machine learning, IDS

1 INTRODUCTION

Smart devices such as smart-speakers and smart light bulbs are appearing increasingly frequently in daily life. These internet connected devices make up the Internet of Things (IoT). IoT-devices develop not only in numbers, but also in complexity. One of the hazards of this development is that a malfunction could have severe consequences, such as a road accident due to a malfunctioning self-driving car. Therefore, the security of IoT devices becomes more and more important. One of the challenges in IoT security is that preventing different attacks requires different strategies [5]. Also, the IoT network is only perfectly secure if all devices on it are secure, due to the fact that poorly secured IoT devices can be used in for example a DDoS attack. This is theoretically appealing, but not realistic. Machine learning (ML) models have been trained which achieve an accuracy score of over 99.9% on the task of detecting anomalies in IoT network traffic flows on the IoT-23 dataset [15]. One of the studies done on the IoT-23 dataset is by Kumari et al. [10]. This dataset consists of the network flow of 23 devices targeted by a successful malware attack [13]. These models are trained on the specific types of attacks in this database, ranging from Distributed Denial of Service (DDoS) to horizontal port scans. Despite the high accuracy achieved by these machine learning models, we cannot assume that over 99.5% of all malicious attacks will be stopped. We can only assume that 99.5% of attacks in the dataset will be stopped. New types of attacks are quickly developed, and little research has been done on how to detect previously unknown attacks. In this research, machine learning models will be trained on manipulated training data from the IoT-23 dataset to simulate the encounter of the model with a new style of attack on an IoT network.

TScIT 41, July 5, 2024, Enschede, The Netherlands

2 CONTRIBUTION

Although there have been many models trained to intercept malicious network flows, the study of detecting new types of attacks against the IoT-network is not sufficient for the safety priority. Additionally, the currently existing ML models have not been tested sufficiently on new types of attack. These ML models have a high performance on the IoT-23 dataset, but have not been tested on unknown attacks in the training data. This paper will analyse the capability of different ML models to detect and prevent new types of attack by both training its own models on the IoT-23 dataset and testing existing ML models on unknown attacks in the training data. This will contribute to the currently available research since little research about this topic exists, while the need for new attack detection increases daily.

2.1 Research Question

What is the impact of excluding specific types of attack from the training data on the performance of machine learning models for anomaly detection in IoT networks using the IoT-23 dataset, and how can we use this knowledge to safeguard against new attacks in the future?

3 RELATED WORK

Nguyen et al. argue there are currently three big challenges in the safeguarding of IoT networks against DDoS attacks [12]. These challenges lie in the limited existence of capable datasets to train an Intrusion Detection System (IDS), the inability to detect new attacks without training data, and the possibility for attackers to use adversarial attacks. This relates to this research since launching a DDoS attack requires access to many IoT-devices, which can be achieved by the attacks described in this research.

3.1 Related work on anomaly detection on new attacks

Several studies have attempted to solve the second challenge - the ability to detect new attacks without training data - using machine learning [2, 7–9, 11, 17, 19]. However, common problems occurred, such as a too large manipulation of training data [7, 9], not suitable dataset [19], poor precision [11], not training on truly new attacks [2, 9, 17], or unclear results [8]. A useful review covering the findings of these studies is [1]. This review covered 92 experiments from 11 studies and concluded a classification error rate of 50.09% which shows the need for new approaches and better methods. These studies can be used for understanding what problems arise when defending against an unknown attack. However, none of this research has been trained on the IoT-23 dataset.

3.2 Related work on anomaly detection using the IoT-23 dataset

On the subject of anomaly detection in the IoT-23 dataset recent studies have shown a successful anomaly detection rate of > 99% [3, 18]. These studies can be used for better understanding of the different aspects needed for successful anomaly detection. A useful review is [15]. However, this research does not discuss performance on new attacks.

4 METHODS OF RESEARCH

Several steps had to be taken to satisfy the research question. Firstly, literature research has been conducted to gain understanding of which ML models perform well for anomaly detection in IoT network flows. Secondly, literature research has been done to gain understanding of the different steps taken on detecting new attacks without training data, and what ML models perform well on this task. After reviewing the results, five different ML models have been chosen that performed well in previous research, or have potential to perform well in this research. The models include two deep learning models. These ML models will be trained and tested on data of the IoT-23 dataset. The results for accuracy, recall and precision have been logged. For each of the five ML models, an instance will be trained on the dataset excluding one type of attack. This is repeated for each attack. Experiments have been conducted to see the effects of excluding different attacks on the results of accuracy, recall and precision. After having trained all five ML models, we can simulate a new type of attack by testing the model against the excluded type of attack from the training data. Since the models have never seen this type of attack before, the result of these tests will be valuable to the understanding of countering new attacks, even though the attacks used in this research are not truly new attacks. Lastly, an analysis of the results of the five ML models on different types of attack has been done to see what models perform the best against previously unseen attacks. This will answer the research question.

5 DATASET

The dataset used for this research is the IoT-23 dataset [13]. This dataset consists of captured network traffic of 20 malware-infected IoT-devices, as well as captured network traffic of three non-malware-infected devices called 'honeypot devices'.[1] The captured data is logged in a Wireshark .pcap file as well as provided conn.log.labeled files, in which all network entries are labeled. In this research, only the conn.log.labeled files are used, since these files contain the relevant information in a convenient format. Each of these files contain data captured over a maximum duration of 24 hours, ranging from 238 to 73,568,982 network flows. In total the dataset consists of 294,449,255 malicious entries, and 30,858,735 benign entries. The malicious entries are subdivided in the labels in Table 2.

Feature	Description	Type	Selected
ts	Timestamp of the capture	int	
uid	Unique ID	str	
id_orig.h	IP-address of the sender	str	
id_orig.p	Port used by the sender	int	
id_resp.h	IP-address of the receiver	str	
id_resp.p	Port of the receiver	int	✓
proto	Network protocol used	str	✓
service	Application protocol used	str	✓
duration	Total time of the interaction	float	✓
orig_bytes	Amount of bytes sent to the device	int	✓
resp_bytes	Amount of bytes sent by the device	int	✓
conn_state	State of the connection	str	✓
local_orig	The connection originated locally	bool	✓
local_resp	The response originated locally	bool	
missed_bytes	Number of missed bytes	int	
history	History of connection state	str	
orig_pkts	Packets being sent to the device	int	✓
orig_ip_bytes	Bytes being sent to the device	int	✓
resp_pkts	Packets being sent from the device	int	✓
resp_ip_bytes	Bytes being sent from the device	int	✓
tunnel_parents	Connection ID, if tunneling was used	str	
label	Label of the capture	str	✓
detailed_label	Detailed label of the capture	str	✓

Table 1. Features of the IoT-23 dataset

Label	Type	Amount	Category
Benign	Benign	30,858,735	0
C&C-PartOfAHorizontalPortScan	Malicious	888	1
PartOfAHorizontalPortScan	Malicious	213,852,924	1
PartOfAHorizontalPortScan-Attack	Malicious	5	1
C&C	Malicious	21,995	2
C&C-FileDownload	Malicious	53	2
C&C-Mirai	Malicious	2	3
C&C-HeartBeat	Malicious	33,673	4
C&C-HeartBeat-Attack	Malicious	834	4
C&C-HeartBeat-FileDownload	Malicious	11	4
C&C-Torii	Malicious	30	5
DDoS	Malicious	19,538,713	6
FileDownload	Malicious	18	7
Okiru	Malicious	47,381,241	8
Okiru-Attack	Malicious	13,609,470	8
Attack	Malicious	9,398	9

Table 2. Labels in IoT-23 dataset

5.1 Data Preparation

First, the separate conn.log.labeled files were converted to CSV-files. Due to memory limitations, larger files had to be split into multiple parts to allow for the creation of the CSV-files. In the original dataset, all missing values were replaced with '-', except for the IP-addresses, where missing values were replaced with '::'. For this research, these missing values were replaced with -1, to ensure the possibility for models to train on the data, as well as have a unique missing data value. Next, the created CSV-files were iterated over and placed in CSV-files based on the *detailed-label*

[1] In this dataset, honeypot devices are not used to catch intrusion attempts. Instead, the honeypot devices in this dataset are only used to simulate normal network traffic.

column, following the numbering in Table 2. Attacks that shared characteristics were grouped under the same number, resulting in ten CSV-files numbered 0-9. This step was an important preparation for the testing phase, saving computational time when splitting data for train/test set.

5.2 Feature Selection

The dataset provides 23 features, as shown in Table 1. Firstly, features that are useful for training on the dataset but have no use in real-life scenarios had to be removed. Using *id_orig.h* is not recommended in ML algorithms, since the model will over-fit on the training data. There are interesting use cases for the IP-addresses, such as replacing the columns with a boolean column whether the sender and receiver IP-address are from the same country. However, since this research focuses more on the network packets, the features for IP-addresses were removed.

Secondly, features that had too few entries were removed. It was rare for these features to have an impact on the classification of the model, and therefore only increased the computational time needed. This was the case for the features *missed_bytes*, *tunnel_parents* and *local_resp*.

Lastly, features that had no impact on the classification of the models had to be removed. This was done by calculating the SHAP values of the features using the python library *shap*. This library returns a figure for a sample of the dataset which shows the importance of each feature on that sample. By running the library on the five models, non-impactful features could be removed. The selected training features are listed in Table 1.[2]

5.3 Data Scaling

In an ideal world, all attacks would have the same number of instances in training and testing. However, some attacks are oversampled and some attacks are undersampled in the IoT-23 dataset. This research did not have the time or resources to create more samples of the undersampled data, and will therefore work with a scaled down dataset of the oversampled data, containing undersampled attacks. Due to hardware limitations, the data had to be scaled to allow efficient training on forest and ANN models. Training a tree model in batches is not recommended, as new batches will only extend the previous branches, instead of modifying them. Also, the imbalance in data for different attacks is clear from Table 2. To solve both the imbalance of data and the computational time for training, attacks with more than 10,000 instances were randomly sampled. The benign data was also scaled down to 10,000 instances. To make sure that all attacks were included, less represented attacks in categories 1, 2, and 4 were included.

6 DATA ANALYSIS

6.1 Attacks

To understand the difference in accuracy among the different attacks, it is important to understand how each attack works, and what the characteristics of each attack is. Therefore, below is a summary of the functionalities and characteristics of the different attacks in the IoT-23 dataset grouped by category.

6.1.1 PartOfAHorizontalPortScan. Port Scanning is an attack method used to prepare for a mass attack. There are two types of port scanning; Horizontal port scanning and vertical port scanning. In horizontal port scanning, the attacker sends requests to the same port on a large amount of devices, to find vulnerabilities. In vertical port scanning, the attacker sends requests to a large amount of different ports on the same device.

In the IoT-23 dataset, the PartOfAHorizontalPortScan attack is simulated by repeating requests many times on the same port of the target device.

6.1.2 C&C. Command & Control attack. Taking over a network of devices after breaching a point of entry. From this entry point, the attacker can take over other devices in the network and get data/ other valuables from it. This server can be used as a center of a botnet. Multiple types of C&C attacks; Botnets, Ransomware, APT, DDoS.

6.1.3 C&C-Mirai. Mirai is a Command & Control attack that focuses on IoT devices using the ARC Processor. This processor has a flaw that enables a Mirai attack to take over the device if the default user/password combination has not been changed. Notably, there are only two instances of the Mirai attack in the IoT-23 dataset.

6.1.4 C&C-HeartBeat. The HeartBeat attack is a type of Command & Control attack that keeps the connection between the host and the infected device active. This is achieved by periodic calls en responses between the host and device, the name of the attacked is derived by this periodic behaviour.

6.1.5 C&C-Torii. The Torii attack is a type of Command & Control attack that is similar to the Mirai attack. However, the Torii attack is better developed to exploit more vulnerabilities and to avoid detection by the host. Torii achieves this by disguising the intrusion as a CSS file download, or by using telnet attacks.[3] Notably, there are only 30 instances of the Torii attack in the IoT-23 dataset.

6.1.6 DDoS. The Distributed Denial Of Service attack is executed by a large number of devices to overflow the network capacity of the targeted device or server. To launch a DDoS attack, the attacker needs to have access to many malware infected devices under their control. By having all these devices repeatedly requesting information from the target, the target is likely to go out of service. In the IoT-23 dataset, the network flows labeled as a DDoS attack are the malware infected device performing a DDoS attack. A DDoS attack can be recognized by many requests to the same target, as well as the requests being sent on a regular basis.

6.1.7 FileDownload. The FileDownload is an attack where instead of data, a file is sent to the target device. Notably, there are only 18 instances of the FileDownload attack in the IoT-23 dataset.

6.1.8 Okiru. The Okiru attack is a type of Command & Control attack that has characteristics of the Mirai attack. The Okiru attack

[2]The feature *detailed-label* is used to know how to split train/test values. However, since we are doing binary classification, this feature is not used for training.

[3]A Telnet attack is a technique that uses the Telnet executable on an older Windows OS. This is used for connecting to a TCP port on a device.

also focuses on devices with the ARC processor. This attack is different from the Mirai attack since Okiru uses different exploits to intrude their target device.

6.1.9 Attack. The label Attack is used on network packets that can not be identified as a known attack or as benign network traffic. Therefore the Attack label contains many different instances with different characteristics. This group of instances is useful for this research because it is the only labeled group that does not have a shared characteristic. Therefore, it will behave differently in the testing phase.

6.2 Models

This research will be performed on five classification models. These were selected from well performing models in previously done research. Both the Random Forest model [18] and Support Vector Machine model [3] performed well with an accuracy of 99.5% and 96.7% respectively. To increase the range of different properties of machine learning techniques to enhance the results of this study, the Histogram Gradient Boosting model, and Recurrent Neural Network model were chosen.

6.2.1 Random Forest. A decision tree is used for classification by creating branches based on difference in features. Once the tree has been traversed, the model will predict the outcome of an instance. The Random Forest (RF) classifier uses the average of a number of different decision trees to predict the class variable [4]. The advantage of using many decision trees is that it protects against over fitting. Selecting correct features is important for RF classifiers, since they build decision trees based on a random sub selection of features, resulting in less efficiency if features do not impact the result. The disadvantage of the Random Forest classifier is the relatively high computational time, which disallowed training on more data.

6.2.2 Histogram Gradient Boosting. Instead of using random decision trees, each decision tree will correct the errors of the previous tree in a Histogram Gradient Boosting model (HGB). The classifier uses a histogram to check for feature importance, and to select the useful features for later trees. This classifier is memory-efficient due to this feature selection, and computes fast on large amounts of data.

6.2.3 Support Vector Machine. A Support Vector Machine (SVM) uses statistical analysis to optimize a hyperplane, the space in between classes in data. The algorithm finds this optimal plane by using so called 'support vectors', which are the data points closest to the hyperplane [6]. A SVM works well with different data types. However, it requires a large amount of memory, as does it have a high computational time.

6.2.4 Recurrent Neural Network. A Recurrent Neural Network (RNN) is a deep learning model that is used to train on sequential data. The model stores an amount of internal data from previous inputs to more accurately predict the next input [16]. Since the data in this experiment will be randomized, there will be less sequential information for the model to use. However, finding the optimal value for the amount of previous inputs saved in the machine will improve performance.

6.2.5 Multilayer Perceptron. A Multilayer Perceptron (MLP) is a deep learning model that consists of multiple layers of nodes connected to each other [14]. The model processes input data independently, making it suitable for the randomized data used in this experiment. These hidden layers make it possible for the model to discover complex feature connections and will therefore make the model more accurate.

7 RESULTS

7.1 Experiment Environment

The models were trained on a Dell PowerEdge R7515, enabled by the Jupyter server of the University of Twente. This CPU enables the use of 64 cores, 128 threads and 1024GB of memory.

The HGB, SVM, and RF models were implemented using the *scikit-learn* library in Python.

For the Recurrent Neural Network, a three-layer model was used. The input layer consisted of an Embedding Layer with 14 input dimensions, and 64 output dimensions. The Long Short-Term Memory Layer had 128 units. The last layer was a Dense Output Layer with 2 units and the Sigmoid activation function.

For the Multilayer Perceptron, a four-layer model was used. The input layer consisted of a Dense Layer with 14 units. The first hidden layer consisted of a Dense Layer with 28 units. The second hidden layer consisted of a Dense Layer with 56 units. The output layer consisted of a Dense Layer with 2 units, and the Sigmoid activation function. For both hidden layers, the ReLU activation function was used. Both Deep Learning models were implemented using the *keras* library and trained for 30 epochs.

Each of the five ML models was trained on eight attack categories and 80% of the benign data, before being tested on the excluded attack category and the remaining 20% of benign data. Both the benign data from malware infected devices as well as honeypot devices was split separately. This approach resulted in 45 models total. The Accuracy, Benign Accuracy, Precision, Recall and F1-Score of each model is logged in Appendix A[4].

7.2 General attack analysis

To analyze how the different attack categories behave in the models, we will look at the recall metric. The importance is the % of malicious instances correctly detected out off all malicious instances, therefore we use the following formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{1}$$

[4]Results denoted with * imply a low amount of samples.

	HGB	RF	SVM	RNN	MLP
1	0.1411	0.4080	0.4063	0.5399	0.4141
2	0.1961	0.2551	0.3919	0.9976	0.4134
3*	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.6453	0.6542	0.6449	0.9472	0.6970
5*	0.0000	0.9667	0.2333	0.9667	0.2333
6	0.9375	0.7954	0.9986	0.9970	0.9417
7*	1.0000	1.0000	1.0000	1.0000	1.0000
8	0.2195	0.9996	1.0000	1.0000	1.0000
9	0.8849	0.2924	0.9999	0.2897	0.9900

Table 3. Recall of different models

The table shows correlations between different attack categories. Firstly, we notice that the models have an overall low performance when the test set contains attack 1. The PartOfAHorizontalPortScan does not have shared characteristics with other attacks from the dataset. Therefore, the models classified many False Negatives. The RNN model performed best, with a recall of approximately 0.54.

Secondly, we notice there is a correlation between the Command & Control attacks, consisting of category 2, 3, 4 and 5. Category 3 only has 2 instances, which all models correctly identified. However, the HGB, RF and SVM model all struggled with the other C&C categories, scoring 0.20-0.39 and approximately 0.65 for categories 2 and 4 respectively. The RNN model scores high on these C&C attacks, approximately 0.97. The MLP model struggles with the C&C attacks, scoring low recall. However, this model scores excellent on attack category 6-9.

All models except HGB perform exceptionally on category 8. We see a notable difference between the performance of the HGB, SVM and MLP model, and the RF and RNN model on category 9. As said before, category 9 is all the attacks that could not be labeled, and therefore contain a broad variance of attacks.

7.3 Model result analysis

Weighted averages for each metric are used to analyse model performance. The results are displayed in this table:

	HGB	RF	SVM	RNN	MLP
Accuracy	0.6064	0.6601	0.7310	**0.8173**	0.7828
Benign Accuracy	0.9962	**0.9989**	0.6673	0.8844	0.9568
Precision	0.9952	**0.9993**	0.9101	0.9597	0.9835
Recall	0.5001	0.5706	0.7375	**0.8005**	0.7400
F1-Score	0.6005	0.6872	0.7905	**0.8436**	0.8191

Table 4. Weighted averages for each model

7.3.1 Histogram Gradient Boosting. The Histogram Gradient Boosting model has the worst performance out of all tested models. The weighted accuracy is approximately 0.61. Although the model has an extremely high benign accuracy of > 0.99, the model has a poor recall that indicates many false positives. Due to the tree structural nature of the model that includes feature selection and regularization, it performs extremely well on benign data but not on malicious data. Therefore, the F1-score of 0.60 indicates poor generalization.

7.3.2 Random Forest. The Random Forest model scored a moderate weighted accuracy of approximately 0.66. The model performed well on low sample categories, and performed worse on high sample categories. The model excelled at Benign Accuracy, scoring > 0.99. However, the recall score of 0.57 indicates many false positives. RF also has a tree structural nature with feature selection and regularization. Overall the RF model has a solid performance with a F1-score of 0.69. The model shows a capability to generalize, though accuracies differ per attack.

7.3.3 Support Vector Machine. The Support Vector Machine model achieved a weighted accuracy of approximately 0.73, showing its relatively good performance across different attack types. However, the model had a significantly lower benign accuracy of approximately 0.67. Despite this, the SVM scored high in precision, achieving a score of approximately 0.91, which suggests that when it does classify an attack, it is likely to be correct. The recall score of approximately 0.74 shows that the model successfully identifies a large portion of attacks. This balance between precision and recall results in a solid F1-score of approximately 0.79. Overall, the SVM model shows potential, although the low benign accuracy will result in problems in a real life scenario.

7.3.4 Recurrent Neural Network. The Recurrent Neural Network model achieved the highest weighted accuracy of approximately 0.82 among the models, highlighting its effectiveness in handling different attacks. It also scored a high benign accuracy of approximately 0.88. However, this is not as high as the HGB and RF model. The precision of approximately 0.96 suggests that the RNN model is reliable when it identifies an attack. The recall score of approximately 0.80 shows that it detects a large proportion of actual attacks. With a F1-score of approximately 0.84, the RNN model shows promise for generalization, which is also noticeable by the accuracy and benign accuracy being relatively similar. One important notice is that the RNN model performed the worst of all models on category 9, even though the model has a large generalization capability. Also, the model performed poorly on low-sampled categories.

7.3.5 Multilayer Perceptron. The Multilayer Perceptron model achieved scores similar to the RNN model. It performed worse on attack category 2, while outperforming the RNN model on attack category 9. The MLP model showed excellent benign accuracy, approximately 0.96. The precision of approximately 0.98 shows that the model is likely correct when it classifies an attack. However, the recall of approximately 0.74 is a bit lower compared to the RNN model, showing that the model classified more false negatives. the MLP model has the second highest F1-score, of approximately 0.82, therefore the model can generalize well. Importantly, the MLP model scored the highest accuracy on attack category 9. The nature of attack category 9 shows an even higher potential for generalization than the RNN model.

8 DISCUSSION

Among the tested models, the RNN model performed the best with the highest weighted accuracy, recall, and F1-score. The large LSTM layer enabled the model to capture feature dependencies in the data. The RNN model has even more potential to perform in real life

scenarios, if the model is able to learn from real time sequential data. The MLP model performed slightly worse, even despite its great performance on attack category 9. This performance does show generalizing capability, making it perhaps more suited than the RNN model. The performance of the MLP model could be increased even more by better hyperparameter tuning.

Both of these performances were based on the weighted average. However, the RF model outperformed other models on both benign accuracy, precision and F1-score on non-weighted averages. This shows that the RF model might be a solid option detecting new attacks, although more research on this is needed due to insufficient sample size of some attacks. Both the HGB and SVM model had issues that resulted in low performance. The HGB model had a low recall and low F1-score, while the SVM model had a low benign accuracy and low precision.

9 CONCLUSIONS

To answer the research question, the impact of excluding specific types of attack from the training data has a negative influence on machine learning models that are bad at generalizing, while having less impact on machine learning models that are good at generalizing. This will especially impact model performance in real life, where the attacks are not limited to the nine categories used in this research. It is important to train models on attacks with different characteristics, to ensure the best possible protection against malicious network traffic.

9.1 Future Research

For future research, this method can be extended to include more types of attacks outside of the IoT-23 dataset. Also, to further improve the results of this method, the number of underrepresented attacks could be increased to ensure validity. The results can further be improved by including more data for model training, which was not enabled in this research due to computational power.

Instead of simulating a new attack by excluding data from the training set, it would have been better to compute a new type of attack specifically designed to test the generalizing capability of models. This is not only more realistic, but will also ensure more data for model training.

Additionally, experimentation on this method by using different machine learning and deep learning models with more accurate hyperparameter tuning could further increase accuracy, as well as combining the average classification results of multiple models.

One of the improvements is to modify the Multilayer Perceptron network, since a MLP model should theoretically work better on non-sequential data than a RNN. In this research, the MLP model showed more promise for generalization. However, this did not reflect on the metrics.

By addressing these points of improvement, future research can improve the results of the method described in this study.

ACKNOWLEDGMENTS

During the preparation of this work the author used OpenAI ChatGPT-4 in order to resolve errors in code execution for both data preparation and model training. After using this service, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

REFERENCES

[1] Malek Al-Zewairi, Sufyan Almajali, and Moussa Ayyash. 2020. Unknown Security Attack Detection Using Shallow and Deep ANN Classifiers. *Electronics* 9, 12 (2020). https://doi.org/10.3390/electronics9122006

[2] F. Amato, F. Moscato, Fatos Xhafa Xhafa, and E. Vivenzio. 2018. Smart Intrusion Detection with Expert Systems. In *Lecture notes on data engineering and communications technologies*. 148–159.

[3] Maria Balega, Waleed Farag, Soundararajan Ezekiel, Xin-Wen Wu, Alicia Deak, and Zaryn Good. 2022. IoT Anomaly Detection Using a Multitude of Machine Learning Algorithms. In *2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. 1–7. https://doi.org/10.1109/AIPR57179.2022.10092209

[4] Leo Breiman. 2001. *Random Forests.* Kluwer Academic Publishers.

[5] Andrew Churcher, Rehmat Ullah, Jawad Ahmad, Sadaqat ur Rehman, Fawad Masood, Mandar Gogate, Fehaid Alqahtani, Boubakr Nour, and William J. Buchanan. 2021. An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. *Sensors* 21, 2 (2021). https://doi.org/10.3390/s21020446

[6] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications* 13, 4 (1998), 18–28. https://doi.org/10.1109/5254.708428

[7] Weirong Jiang and V. Prasanna. 2014. *Hardware Techniques for High-Performance Network Intrusion Detection.* 233–256. https://doi.org/10.1201/b16390-13

[8] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman, and Ammar Alazab. 2020. Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine. *Electronics* 9, 1 (2020). https://doi.org/10.3390/electronics9010173

[9] Przemysław Kukiełka and Zbigniew Kotulski. 2010. Analysis of neural networks usage for detection of a new attack in IDS. *Annales UMCS, Informatica* 10 (01 2010), 51–59. https://doi.org/10.2478/v10065-010-0035-7

[10] Priyanka Kumari, Veenu Mangat, and Anshul Singh. 2023. Comparative Analysis of State-of-the-Art Attack Detection Models. 1–7. https://doi.org/10.1109/ICCCNT56998.2023.10306428

[11] Jorge Meira, Rui Andrade, Isabel Praça, João Carneiro, and Goreti Marreiros. 2019. *Comparative Results with Unsupervised Techniques in Cyber Attack Novelty Detection.* 103–112. https://doi.org/10.1007/978-3-030-01746-0_12

[12] Xuan-Ha Nguyen and Kim-Hung Le. 2023. Robust detection of unknown DoS/DDoS attacks in IoT networks using a hybrid learning model. *Internet of Things* 23 (2023), 100851. https://doi.org/10.1016/j.iot.2023.100851

[13] Agustin Parmisano, Sebastian Garcia, and M. J. E. 2018. IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic. *Stratosphere IPS* (2018).

[14] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems* 8, 7 (2009), 579–588.

[15] Saida Hafsa Rafique, Amira Abdallah, Nura Shifa Musa, and Thangavel Murugan. 2024. Machine Learning and Deep Learning Techniques for Internet of Things Network Anomaly Detection—Current Research Trends. *Sensors* 24, 6 (2024). https://doi.org/10.3390/s24061968

[16] Hojjat Salehinejad, Julianne Baarbe, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. 2018. Recent Advances in Recurrent Neural Networks. *CoRR* abs/1801.01078 (2018). arXiv:1801.01078 http://arxiv.org/abs/1801.01078

[17] Pranesh Santikellur, Tahreem Haque, Malek Al-Zewairi, and Rajat Chakraborty. 2019. Optimized Multi - Layer Hierarchical Network Intrusion Detection System with Genetic Algorithms. 1–7. https://doi.org/10.1109/ICTCS.2019.8923067

[18] Nicolas-Alin Stoian. 2020. Machine Learning for anomaly detection in IoT networks : Malware analysis on the IoT-23 data set. essay.utwente.nl.

[19] Tuan Anh Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, Mounir Ghogho, and Fadi El Moussa. 2020. DeepIDS: Deep Learning Approach for Intrusion Detection in Software Defined Networking. *Electronics* 9, 9 (2020). https://doi.org/10.3390/electronics9091533

A APPENDIX: MODEL RESULTS

A.1 Accuracy

	HGB	RF	SVM	RNN	MLP
1	0.3035	0.5213	0.4377	0.6034	0.5185
2	0.3707	0.4212	0.4694	0.9722	0.5240
3*	0.9975	0.9992	0.6924	0.9119	0.9473
4	0.7122	0.7203	0.6549	0.9322	0.7470
5*	0.9850	0.9983	0.6847	0.9063	0.9383
6	0.9494	0.8344	0.9369	0.9779	0.9431
7*	0.9983	0.9987	0.6950	0.8756	0.9585
8	0.3688	0.9994	0.9519	0.9787	0.9920
9	0.9528	0.4496	0.9487	0.4145	0.9836

Table 5. Accuracies of different models

A.2 Benign Accuracy

	HGB	RF	SVM	RNN	MLP
1	0.9886	0.9996	0.6072	0.8713	0.9591
2	0.9983	0.9983	0.7100	0.8722	0.9582
3*	0.9975	0.9992	0.6906	0.9118	0.9473
4	0.9945	0.9992	0.7073	0.8688	0.9582
5*	0.9975	0.9987	0.6906	0.9055	0.9473
6	0.9996	0.9987	0.6022	0.8975	0.9489
7*	0.9983	0.9987	0.6906	0.8747	0.9582
8	0.9987	0.9987	0.6906	0.8890	0.9582
9	0.9975	0.9987	0.6873	0.9093	0.9582

Table 6. Benign Accuracies of different models

A.3 Precision

	HGB	RF	SVM	RNN	MLP
1	0.9812	0.9998	0.8482	0.9465	0.9771
2	0.9980	0.9984	0.8801	0.9684	0.9749
3*	0.2500	0.5000	0.0035	0.0095	0.0157
4	0.9980	0.9997	0.9229	0.9682	0.9860
5*	0.0000	0.9063	0.0121	0.1146	0.0530
6	0.9999	0.9996	0.9315	0.9762	0.9884
7*	0.8182	0.8571	0.0306	0.0571	0.1538
8	0.9986	0.9997	0.9461	0.9744	0.9902
9	0.9993	0.9990	0.9422	0.9268	0.9895

Table 7. Precision of different models

A.4 Recall

	HGB	RF	SVM	RNN	MLP
1	0.1411	0.4080	0.4063	0.5399	0.4141
2	0.1961	0.2551	0.3919	0.9976	0.4134
3*	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.6453	0.6542	0.6449	0.9472	0.6970
5*	0.0000	0.9667	0.2333	0.9667	0.2333
6	0.9375	0.7954	0.9986	0.9970	0.9417
7*	1.0000	1.0000	1.0000	1.0000	1.0000
8	0.2195	0.9996	1.0000	1.0000	1.0000
9	0.8849	0.2924	0.9999	0.2897	0.9900

Table 8. Recall of different models

A.5 F1-Score

	HGB	RF	SVM	RNN	MLP
1	0.2467	0.5795	0.5494	0.6876	0.5817
2	0.3278	0.4064	0.5423	0.9828	0.5806
3*	0.4000	0.6667	0.0070	0.0188	0.0309
4	0.7838	0.7840	0.7593	0.9576	0.8167
5*	0.0000	0.9355	0.0230	0.2049	0.0864
6	0.9677	0.8859	0.9639	0.9865	0.9645
7*	0.9000	0.9231	0.0594	0.1080	0.2666
8	0.3599	0.9996	0.9723	0.9870	0.9951
9	0.9386	0.4524	0.9702	0.4414	0.9897

Table 9. F1-Score of different models

A.6 Averages

	HGB	RF	SVM	RNN	MLP
Accuracy	0.7376	0.7714	0.7191	0.8414	0.8391
Benign Accuracy	0.9967	0.9989	0.6752	0.8889	0.9548
Precision	0.7826	0.9177	0.6130	0.6602	0.6810
Recall	0.5583	0.7079	0.7417	0.8598	0.7433
F1-Score	0.5472	0.7370	0.5385	0.5972	0.5902

Table 10. Averages for each model

A.7 Weighted Averages

	HGB	RF	SVM	RNN	MLP
Accuracy	0.6064	0.6601	0.7310	0.8173	0.7828
Benign Accuracy	0.9962	0.9989	0.6673	0.8844	0.9568
Precision	0.9952	0.9993	0.9101	0.9597	0.9835
Recall	0.5001	0.5706	0.7375	0.8005	0.7400
F1-Score	0.6005	0.6872	0.7905	0.8436	0.8191

Table 11. Weighted averages for each model