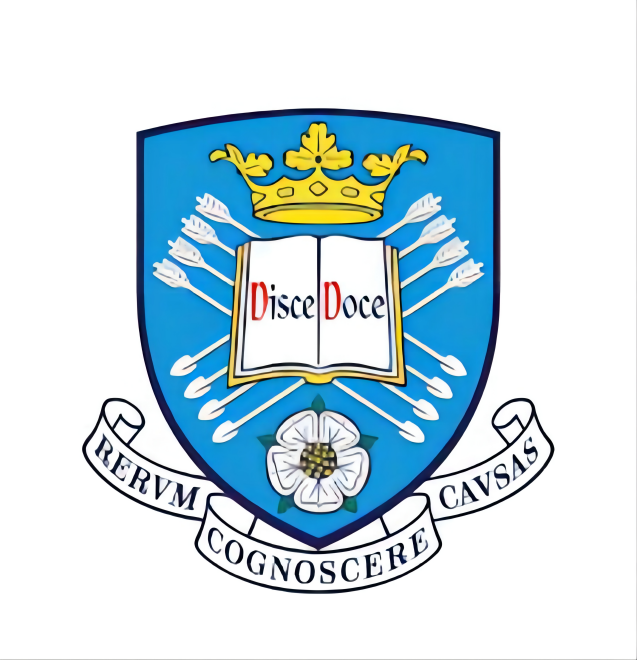
University of Sheffield

**Malicious Endpoint Detection & Response using ML**



Shangyuan Liu

*Supervisor*: Dr. Olakunle Olayinka

A report submitted in partial fulfilment of the requirements

for the degree of MSc Cybersecurity and Artificial Intelligence

*in the*

Department of Computer Science

May 11, 2022

**Chapter 1**

**Introduction**

计算机网络的安全是一个国际化的问题，每年全球因计算机网络的安全系统被破坏而造成的经济损失达上千亿美元。当商户、银行与其他商业与金融机构在电子商务热潮中纷纷进入Internet，以政府上网为标志的数字政府使国家机关与Internet互联。通过Internet 实现包括个人、企业与政府的全社会信息共享已逐步成为现实。随着网络应用范围的不断扩大，对网络的各类攻击与破坏也与日俱增。无论政府、商务，还是金融、媒体的网站都在不同程度上受到入侵与破坏。网络安全已成为国家与国防安全的重要组成部分，同时也是国家网络经济发展的关键。对入侵攻击的检测与防范、保障计算机系统、网络系统及整个信息基础设施的安全已经成为刻不容缓的重要课题。

**Chapter 2**

**Literature Review**

**2.1 Background**

As technology rapidly develops and the information age expands, the technology areas such as 5G, cloud computing, the Internet of Things (IoT) and artificial intelligence have been brought about significant innovation. The number of global IoT connections is expected to reach 14.4 billion approximately by the end of 2022 and the number of IoT devices will rise to around 27 billion in 2025 [1]. However, the endpoint environment, comprising various devices including the IoT, smartphones and laptops etc, is being exposed to massive cyber attacks such as ransomware, malware, phishing, DDoS, and botnets [2]. According to research studies, up to 90% of successful cyber attacks and 70% of successful data leakage incidents emanate from endpoint devices [3]. Moreover, the endpoint environment also has been challenged by many unknown malicious attacks. Most significantly, traditional endpoint security tools and approaches are ineffective at detecting or eliminating advanced threats. As a result, these threats can lurk on the network for months, collecting data and identifying vulnerabilities to prepare for ransomware attacks and zero-day exploit attacks.

**2.****2 Endpoint** **Detection and Response**

In order to mitigate malicious cyber threats, [Anton Chuvakin](https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/" \t "https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/_blank) suggested endpoint detection and response (EDR) [4], which is a solution to fortify endpoint security that will continuously collect endpoint data (computers, servers, mobile devices, IoT devices, etc.). And combined with automated rules-based response and analysis capabilities to analyse and detect suspicious cyber activities and threats on endpoints, and automatically respond to secure endpoint devices and IT assets [5].

In terms of the history of endpoint security, traditional anti-virus (AV) is the first generation, which primarily matches known virus databases and signature databases to detect and kill viruses. The endpoint protection platform (EPP) is the second generation. The main function of the EPP system is to fortify endpoint devices' security, including anti-virus, anti-spyware, anti-cyber-attack, anti-phishing, firewall, prevent unauthorized access [8], and also provide data loss protection and data encryption. It is a comprehensive endpoint protection product. While EDR belongs to the third generation, it relies on an accurate indicator of compromise (IOC) threat intelligence information support combined with big data analysis, artificial intelligence, analysis technologies, thus forming a complete endpoint protection system [9]. The emergence of EDR remedies the shortcomings of the traditional EPP, which is not a complete substitute for EPP and AV, but a compliment to each other. In the future, there will be a move towards a more comprehensive Extended Detection and Response (XDR) [10].

EDR technology differs from the previous static defence based on decision boundaries, rules, and strategies. It is a proactive endpoint security protection technology that monitors any security threats and malicious activities by recording endpoint and network events, combining known IOCs, using behaviour analysis and machine learning techniques to monitor any security threats and malicious activities, and automating blocking, forensics, and remediation. Therefore, EDR technology has become a significant role in the overall security protection system in cyberspace because of its sophistication and superiority against unknown threat attacks, zero-day vulnerability attacks, APT attacks, and fileless attacks.

**2.3** **Threats to** **Endpoint Devices**

Due to COVID-19 pandemic, the number of attacks detected in the first quarter of 2020 alone reached 445 million [64], and the attacks against endpoint devices can be classified into the following eight main categories.

**Ransomware**

Ransomware is a type of malware that is deployed on enterprise endpoint devices frequently. And it employs encryption algorithms to encrypt data and files that infected individuals generally are unable to decrypt, and all ransomware requires victims to pay a ransom in exchange for regaining access. Endpoint security includes advanced threat detection to identify and block ransomware before it encrypts data. [11] [12].

**Advanced Persistent Threat**

Advanced Persistent Threat (APT) is a sophisticated, persistent, and multi-stage cyber attack that includes three elements: advanced, long-term, and threatening. Advanced means that sophisticated malware and techniques are used to exploit vulnerabilities in systems. And Long-term refers to the continuous monitoring of a specific target, retaining access to it, and obtaining data from it over time. Threatening emphasises human involvement in planned attacks that target high-value organisations [13]. Successful APT attacks result in significant financial loss or political impact, and therefore APT is currently one of the most serious threats to companies and governments [13][14].

**Malware**

Malware has the ability to access the wireless and sensor network environment in a variety of methods. Once an intrusion is successful, it is more difficult to prevent than TCP/IP networks in terms of transmissibility, invisibility, and destructiveness. It can be difficult to detect and remove such malicious code.

**Phishing**

Phishing describes the use of fraudulent emails or other digital communications to trick users into revealing sensitive information. Basically, phishing attacks account for over half of all cybercrime, and phishing attacks can install malicious ransomware, Trojan horses, etc. on targeted endpoint systems [15]. Therefore, it can result in the loss of large amounts of sensitive private data from endpoint devices and can lead to identity theft, which can cause more serious security incidents [16]. Advanced Endpoint Security solutions integrate email gateways that identify and isolate malicious emails, thereby reducing the risk of users falling into the trap of phishing tactics.

**Botnet**

A botnet comprises a series of hosts infected with malware. Attackers infect a wide range of hosts on the Internet by spreading botnets through various means, forming a botnet. Furthermore, it can be used by the attacker to perform a series of malicious activities, such as distributed denial of service attacks, sending spam, stealing personal information, and performing distributed computing tasks. Such cyber-attacks can result in the crippling of the entire IoT infrastructure network or critical applications, as well as the leakage of a significant amount of confidentiality or personal privacy, and can also commit other criminal activities.

**Port Scanning**

Port scanning is defined as a client sending a corresponding request to a certain range of server ports in order to confirm which ports are available. However, the technique has been used for malicious network activities, where hackers send a set of port scan messages in order to find vulnerabilities within a specific computer port to obtain sensitive information from the user [5]. In response to such attacks, users should implement monitoring of port traffic to prevent attackers from exploiting potential vulnerabilities to gain unauthorized access to the network and commit illegal activities [17].

**Sensor-Based Threats**

Smart IoT devices are vulnerable to sensor-based threats and attacks due to the lack of proper security mechanisms to control the use of sensors by installed applications. By exploiting sensors on smart devices (e.g. accelerometers, gyroscopes, microphones, light sensors, etc.), attackers can extract information from the device, transmit malware to it or trigger malicious activity to compromise it [18].

**Telnet Attacks**

Telnet is a TCP/IP application, as well as a protocol used to connect a local computer to a remote computer. But for an attacker, Telnet is just a tool for remote login. Once an attacker has established a Telnet connection to a remote host, the intruder can use the software and hardware resources on the target host. Due to the continued increase in Tenet-based attacks in recent years, Šemić, H. and Mrdovic, S. proposed an approach for implementing a honeypot for detecting and responding Tenet attacks on IoT devices [19]. Furthermore, IoT honeypots are used to analyse malicious Tenet traffic on IoT devices running on different CPU architectures.

**2.4 Machine Learning-based Detection Algorithm**

The number, variety, speed of spread, and difficulty of detection of malware have increased, which poses a significant challenge to the security of endpoint devices [20]. Therefore, the traditional malware detection techniques that are based on features and rules do not seem to provide the most effective method to protect users from malware. According to Kaur, H., and Tiwari, R. [5], the research identified malware that uses AI algorithms to alter its signature to evade the EDR system. Therefore, research on malicious endpoint detection based on ML and deep learning is a popular research direction in the field of cyber security in recent years. The technology is able to provide excellent solutions in practical applications, not only improving efficiency but also reducing detection costs [21]. In this section, some common ML algorithms are compared that have been proven to detect and respond to malicious endpoints.

**Support Vector Machine (SVM)**

In 1992, the Support Vector Machine (SVM) was introduced by Boser et al. [22], which is a supervised learning model in classification problems and regression analysis. The basic principle of SVM is to dichotomise the training data set correctly and separate the hyperplanes at the maximum geometric interval. In Figure 2.1, is the separating hyperplane.

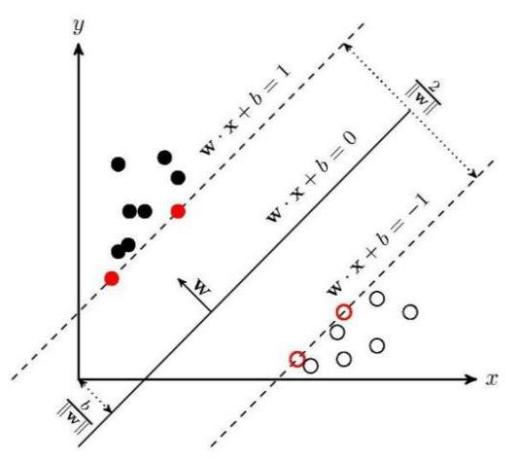


Figure 2.1 Fundamental of Support Vector Machine Architecture

The SVM algorithm was originally designed for binary classification problems [23], making SVM one of the most popular algorithms for malicious endpoint detection. Previous research by Ambwani [28] showed using multi-class SVM classifiers to identify various types of attacks, and the application of SVM in intrusion detection had a good performance. Moreover, SVM has been proven to have a higher potential compared to traditional methods. However, it does not perform well in multi-classification detection, and some questions remain to be addressed.

Furthermore, some authors have suggested [31] a semi-supervised learning mechanism for alarm filters in intrusion detection systems (IDS), which are built using the Reduced Support Vector Machine (RSVM) [30] algorithm to reduce false alarms from NCF instances. The final results show that the use of connection features with supervised and semi-supervised learning techniques successfully reduces false positives by up to 85%, while still maintaining a high attack detection rate. However, the present studies were constrained to the construction of an alert filter using partially annotated data in this approach is not satisfactory.

In order to address the quadratic programming (QP) problem that occurs in the training process of SVMs, the research of Tian et al. [29] has come up with a good solution, which is an AdaBoost.M1+ SVM algorithm based on an integrated multi-classifier and used the Sequential Minimal Optimization (SMO) algorithm to solve the optimization problem of the SVM objective function. Then executing the relevant files generates a log file that provides statistical analysis of API calls in a virtual environment. The SVM model finally achieves 95.2% malware classification accuracy after the weighted average.

The previous research of Ioannou, C. and Vassiliou, V. [32][33] demonstrated an algorithm that the C-Support Vector Machines (SVM) was used to detect network layer attacks on IoT devices. The C-SVM model was trained using benign and malicious local sensor activities and could achieve 100% accuracy in detecting specific malicious attacks. A closer look at the literature however reveals a number of gaps and shortcomings, such as the performance of the method is not satisfactory when evaluating against routing attacks in different network topologies. In addition, the investigators did not compare the experimental results with other models.

**Decision Trees (DT) -** **Random Forest (****RF)**

The decision tree is a non-parametric supervised learning algorithm for classification and regression tasks. It is a tree structure in which each internal node represents a test on an attribute, each branch represents a test output, and each leaf node represents a category as represented in Figure 2.2. Basically, there are three common DT node classifications based on ID3, C4.5, and CART, where ID3 was proposed by Quinlan J.R [24] that based on information entropy and information gain, Quinlan J. R [25] presented the C4.5 algorithm which is an improved version of ID3 based on information gain rate, and Breiman, L. [26] proposed CART which is based on the Gini coefficient.

Tin Kam Ho (1995) proposed the Random Forest algorithm, which further incorporates the selection of random attributes in the training process of decision trees, based on the construction of Bagging integration using decision trees as base learners [27]. As shown in Figure 2.3, when performing the classification task, each decision tree in the forest is decided and classified separately, and a classification result is obtained separately, then RF will take the most classified result as the final result.

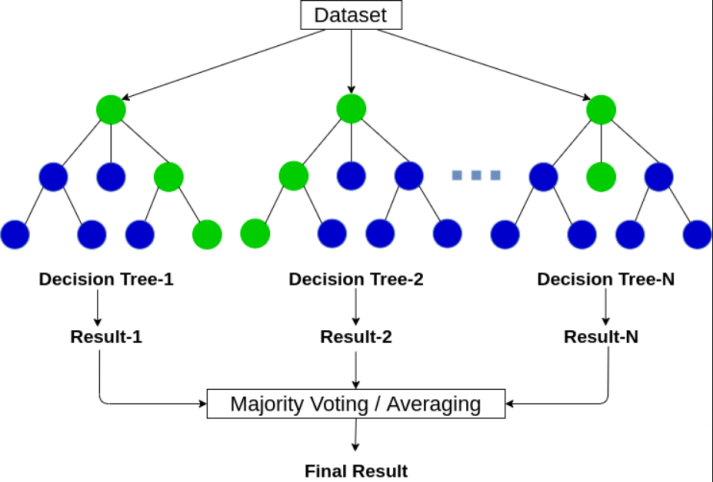


Figure 2.2 Decision Trees Architecture Figure 2.3 Random Forest Architecture

In 2019, Hasan, M. et al. [34] adopted the Distributed Smart Space Orchestration System (DS2OS) to create a virtual IoT environment to capture the required experimental data as well as used the Kaggle dataset provided by Pahl et al. [35]. And several ML algorithms such as DT and RF are applied to identify attacks and anomalous activities against sensors in IoT endpoints. The results of the five-fold cross-validation of the dataset demonstrated the effectiveness of the method, however, in the case of large data and other unknown problems, the method cannot ensure the performance of the classifier.

Furthermore, feature selection is an arguably important question to be addressed. In order to distinguish normal traffic from malicious traffic, it is also possible to identify different types of network attacks, Alothman, Z et al. [36] suggested experiments with multiple classifiers (J48, RF, and MLP) using Pearson’s Correlation Coefficient (PCC) and Relief-F to remove the Bot-IoT dataset some irrelevant features. The final experimental results showed that the RF classifier outperformed other classifiers, but due to data sample imbalance problems, however, the detection rate for specific attack types is not high.

Afterward, Penmatsa, R. K [37] also introduced a method that achieves 99.73% accuracy after a 90% reduction in network traffic features. The method was based on Grey Wolf Optimization (GWO) to optimize features in botnet data without compromising classification accuracy. The GWO optimisation algorithm was also evaluated on SVM, KNN, and DT models respectively, and the GWO algorithm performed best on the DT classifier compared to the other classifiers.

**Logistic regression (LGR)**

Logistic regression is a generalised linear regression analysis model and is frequently used for binary classification. LGR essentially assumes that the data follows this distribution and then uses extreme likelihood estimation to estimate the parameters. It is based on linear regression with a sigmoid activation function, which has the mathematical expression:

IMG_256

In 2019, Ongun, T. et al. [38] leveraged LGR and other ML algorithms to detect the CTU-13 dataset to solve the problem of detecting botnets in network traffic data. In the research, connection-level representation and aggregated traffic statistics feature representation methods were tried for feature extraction. However, the proposed research suffers from limitations due to LGR does not handle the data sample imbalance well compared to random forest and gradient boosting. In order to properly address this question, the data sampling strategies are supposed to be appreciated in data processing stage.

In addition, Machaka, P. et al. [39] investigated a semi-supervised ML model combining LGR and K-Means for detecting DDoS attacks in CPS-IoT in 2021. The One-over-rest training scheme and the bfgs algorithm were utilized to train the LR model for the DAPRA IDS dataset. The use of a semi-supervised ML model resulted in better performance, achieving 100% detection accuracy and 0 false positives.

**Neural Networks**

Neural Networks (NNs) have a structure inspired by the human brain, mimicking the way biological neurons pass signals to each other [41]. Therefore, an NN composed of node layers, containing an input layer, one or more hidden layers, and an output layer, as depicted in Figure 2.4 [40]. Each node is also known as an artificial neuron, which is connected to another node and has associated weights and thresholds. Neural network relies on large amounts of training data and continues to learn to improve its accuracy over time [40], so in general, it consistently outperforms other ML models.

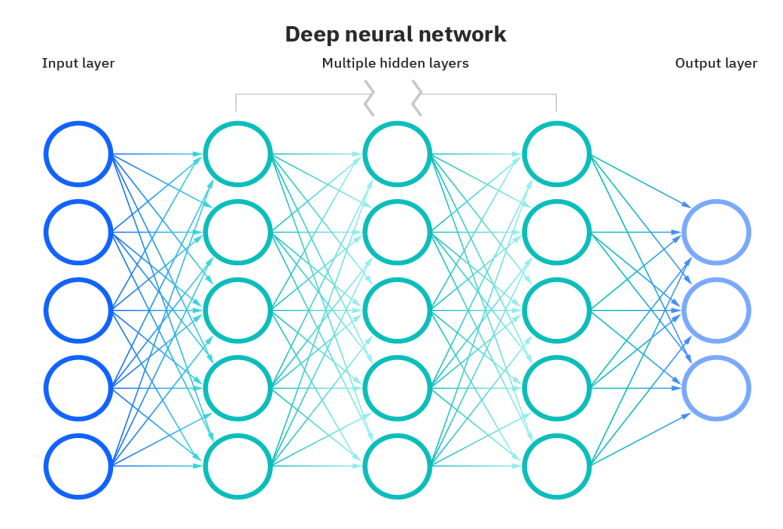


Figure 2.4 Deep Neural Networks Architecture

More formally, NN algorithms have been studied and found to be suitable for big data research and also have become one of the most popular approaches for detecting malicious endpoints and network attacks [42][43]. In 2016, a multilevel intrusion detection approach using NN was proposed by Ji, S.-Y. et al. Discrete Wavelet Transform (DWT) features were used to generate predictive models to detect anomalous behaviour and attacks on network traffic [45]. Similarly,

some authors [46][47] have driven the further development of using DWT combined with ML and data mining methods to construct detection models.

##### Deep Neural Network (DNN) is a comprehensive learning model. Because of its powerful computational capabilities with more hidden layers [49], it can be one of the most widely used learning techniques for detecting malware and endpoints currently. Gao, M. et al. [48] combined a 4-layer DNN and Apriori algorithm to design an IDS. The Apriori association analysis algorithm was applied to compute association rules between discrete features to reduce the false alarm rate. The results showed that the application of association analysis and DNN has good results, but the structure of NN still has much space for optimization.

Nagisetty., A., & Gupta, G.P. [44] offered a system for detecting malicious network traffic in IoT backbone networks using four different deep learning models: Multi-Layer Perceptron (MLP), CNN, DNN, and Autoencoder. The proposed models were evaluated by UNSW-NB15 and NSL-KDD9. The results showed that all models outperformed UNSW-NB15 on the NSL-KDD99 dataset, and DNN had the best performance with an accuracy of 99.24%. This has been previously assessed only to a very limited extent because such datasets do not provide a good representation of the network environment of IoT endpoints.

In respect to real time detection, Lee, W. Y. et al. [50] presented a classification approach for malicious Android applications. A combined model was constructed by stacking RNN and CNN, where CNN and Gated Recurrent Unit (GRU) were used to extract features from the dataset and the RNN was utilized to learn the data features for classification. The approach decreased the training time of the RNN model and achieved a true positive rate of 97.7% with a false positive rate of 0.01.

**2.5 Research Datasets**

In the past two decades of research in the field of cybersecurity, most early studies as well as current work focus on DARPA, KDD-99 CUP, and NSL-KDD datasets [28] [31] [37] [39] [45] [48] [50], which are not only the most widely used datasets but also the benchmarks for evaluating the performance of detection algorithms and models with merits.

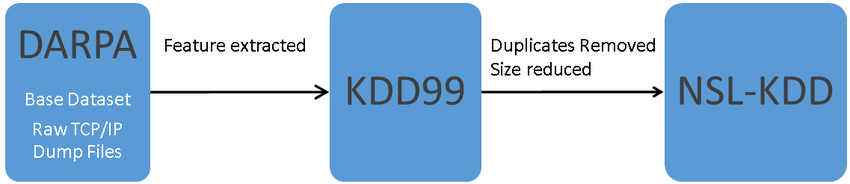


Figure 2.5 The relationship between DARPA, KDD99, and NSL-KDD datasets

Shalaginov et al. [56] proposed an ANN-based model for detecting malicious attacks in the MQTT network environment. And the proposed model was trained and evaluated in the KDD-99 and NSL-KDD datasets, but such datasets do not representative of the IoT system environment under MQTT networks. As shown in Figure 2.5, KDD99 was extracted from DARPA and NSL-KDD was generated from KDD99. However, because of a large amount of data redundancy in the datasets, these datasets have been gradually abandoned because the redundancy problem affects the accuracy of the classifiers [51]. In an age of rapid IoT technological advancement, the conventional Internet traffic datasets are not representative of the traffic found in wireless sensor networks.

The MQTT protocol is one of the most common standards for machine-to-machine communication in the IoT [52]. Therefore, datasets based on the MQTT protocol represent the characteristics of IoT networks better and have been frequently used in IDS and EDR research past years. For instance, the following studies were conducted on MQTT-based devices. In order to address the malicious endpoint problem, Khan, Muhammad Almas et al. [52] in the study used the MQTT-IoT-IDS2020 dataset [53] to construct a DNN-based IDS. The results show that DNN achieves 99% and 98% detection accuracy for binary and multi-class attacks in the feature data models of Uni-Flow and Bi-Flow.

Furthermore, Ullah, I. and Mahmoud, Q.H. [54] created a new dataset based on the MQTT-IoT-IDS2020 dataset and other datasets (MQTTset, IoT-23) and constructed a feedforward neural network (FNN) based model for detecting the malicious endpoints. The new dataset with simplified features reduces the time and computational complexity of training the model and predicting the classification while meeting the high detection accuracy.

MQTTset is another dataset focused on the MQTT protocol, which was proposed and used in IoT networks by Vaccari et al. [55] to classify legitimate traffic and network attacks in MQTT networks. The MQTTset dataset collected and created by different IoT sensors (temperature, motion sensors, etc.) was used in [57] study and the best model of ML was selected to detect attacks in the MQTT-IoT network. The results showed that the trained learning model could secure the IoT environment and achieve an effective detection accuracy of over 99% against attacks in MQTT networks. Idrissi et al. [58] also performed a benchmark test of the proposed algorithm based on the Generative Adversarial Network (GAN) architecture using the MQTTset dataset.

In most relevant datasets, there is not much information included about the botnet scenario. Therefore, Koroniotis, N. et al. [59] suggested a new data set, BoT-IoT, which contains both legitimate and IoT network traffic as well as various types of attacks. Some features with low relevance were removed by using techniques such as feature selection and supervised learning techniques (SVM, RNN, LSTM) were implemented to evaluate the new dataset. The evaluation results showed that both the RNN and LSTM models with an accuracy of over 99.74%. And the dataset with the 10 best features performed better than the full dataset. In 2020, a hybrid algorithm was proposed by Shafiq, M. et al. [60] for BoT-IoT attack traffic identification from several different ML algorithms. And proposed an algorithm to select the most effective ML model based on a bijective soft set approach.

The 23 sub-datasets in the IoT-23 dataset have serious data sample imbalance problems. Therefore, Oha, C.V. et al. [62] created 13 CSV files for each of the 13 labels, containing data from all 23 datasets. The data were random ML selected to generate new datasets to mitigate the underfitting and over-fitting problems associated with sample imbalance. And Abdalgawad, N. et al. [61] presented generative deep learning models such as AAE and BiGAN trained and generated using the IoT-23 dataset. The evaluation results showed that the model outperforms traditional ML algorithms, and the dataset can capture attacks from IoT endpoints effectively.

Liu et al. [63] proposed to use common smart home devices to create the dataset IoT-NI. After data pre-processing, the dataset was applied to five common ML methods, including LGR, SVM, KNN, RF and XGBoost. The accuracy, recall and F1 scores of RF were evaluated to be 100%. Although the RF method provides the highest metric score, it requires the largest amount of computation.

**2.7** **Comparison Table of Past Research Literature**

Table 2.1 summarises the results obtained by applying different ML algorithms on different datasets in recent years. These classifiers and algorithms were trained on the features of the proposed dataset and used to classify different network attacks. It can be seen from Table 2.1 that deep learning-based algorithms perform better than traditional ML methods in detecting malicious endpoints, with an accuracy of 99.9%, but with special feature engineering, some DT and RF classifiers can achieve equally high accuracy rates.

In the last two years of research datasets such as KDD-99, UNSW- NB15 and NSL-KDD are not sufficiently representative of the heterogeneous nature of current IoT networks and their performance in terms of attack diversity among IoT endpoints is fairly limited. Therefore, more and more datasets on IoT are being adopted, such as MQTT-IoT-IDS2020, MQTTset, BoT-IoT, IoT-2 and IoT-NI, which were captured by virtual or realistic IoT sensors. And in order to improve model performance, some datasets were individually designed by the researchers for a certain attack category. From the results, these datasets on IoT are able to accomplish the classification of normal and malicious IoT traffic with high accuracy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Year** | **Dataset** | **Max ACC** | **Max F1** | **REF** |
| Muti-Class SVM Classifiers | 2003 | KDD-99 | 91.67% |  | [28] |
| AdaBoost.M1+ SVM - SMO | 2010 | VET Zoo | 95.2% |  | [29] |
| RSVM | 2010 | KDD-99 | 90.91% |  | [31] |
| C - SVM | 2019 | RMT tool | 75% |  | [32] |
| Decision Tree / Random Forest | 2019 | DS2OS | 99.4% | 0.99 | [34] |
| J48, RF and MLP | 2020 | Bot-IoT | 99.9% |  | [36] |
| GWO - DT | 2021 | KDD-99 | 99.73% |  | [37] |
| Logistic Regression | 2019 | CTU-13 | 99.8% | 0.97 | [38] |
| LGR - K-Means semi-supervised | 2021 | DAPRA IDS | 100% | 1.0 | [39] |
| MLP / CNN / DNN / Autoencoder | 2019 | NB15 / NSL-KDD | 99.24% | 0.9928 | [44] |
| NN - DWT | 2016 | NSL-KDD | 96.67% |  | [45] |
| 4-layer DNN + Apriori | 2020 | NSL-KDD |  | 0.70 | [48] |
| CNN - GRU / RNN | 2019 | NB15 / NSL-KDD | 99.24% |  | [50] |
| DNN | 2021 | MQTT-IoT-IDS2020 | 99.9% |  | [53] |
| FNN | 2022 | MQTT-IoT-IDS2020  MQTTset / IoT-23 | 99.93% | 0.993 | [54] |
| LGR / KNN / RF / SVM / DT | 2022 | MQTTset | 99.89% | 1.0 | [57] |
| GAN | 2022 | MQTTset | 99% |  | [58] |
| SVM / RNN / LSTM | 2019 | BoT-IoT | 99.97% |  | [59] |
| DT / RF / Naïve Bayes / BayesNet | 2020 | BoT-IoT | 99.99% | 1.0 | [60] |
| AAE / BiGAN | 2022 | IoT-23 |  | 0.99 | [61] |
| LR / DT / RF / XGBoost / ANN | 2022 | IoT-23 | 99.99% | 1 | [62] |
| LGR / SVM / KNN / RF / XGBoost | 2020 | IoT-NI | 100% | 100% | [63] |

Table 2.1 Comparison Table of Past Research Literature

**2.8 --- critical discussion**

**2.9 Summary**

In conclusion, this chapter provides a considerable investigation into malicious endpoint detection and response using ML algorithms. In order to implement the detection of malicious traffic in IoT devices, one or more classification detection algorithms are required to analyse a large amount of endpoint data. It is also important to select the appropriate feature selection and data sampling algorithms to reduce data redundancy. Therefore, in previous studies, various ML optimisation algorithms have been proposed as solutions that can improve the performance of classifying malicious endpoints through continuous learning. However, the previous studies cannot be considered conclusive.

Although ML is becoming increasingly used in the field of cybersecurity, it also has some shortcomings. Because some studies have used unreliable datasets, which cannot be used to evaluate malicious endpoint detection frameworks for IoT network environments. Furthermore, some of the research objectives only distinguish between two categories of traffic data: normal and abnormal, without a detailed classification of the abnormal traffic data. Moreover, through literature review, it was found that some authors used Accuracy as a metric to evaluate the proposed model. However, it is more reasonable to use the Recall metric for malicious endpoint detection, as a higher Recall means a higher probability of the actual malicious data being predicted.

One of the tough challenges for all researchers in this domain is extracting relevant and valuable feature information from large-scale IoT traffic data. A variety of traffic features should be collected and analysed to tackle this challenge. And a novel approach is therefore needed for processing data and enabling detection of unknown attacks.

**Chapter 3**

**Requirement and Analysis**

**3.1 Project Overview**

The purpose of this research project is to build a malicious endpoint detection and response system. It is a novel, intelligent and fast proactive defence technology that follows an adaptive security architecture. Furthermore, ML algorithms are used to detect endpoint devices and find malicious activities, both known and unknown threats in a timely manner, and to quickly and intelligently classify and respond accordingly. In addition, the ML model is supposed to demonstrate excellent performance and robustness, as discussed in Section 2. It helps to maintain the accuracy of the determination when the models are confronted with small changes in the input data.

**3.2 System Requirements**

**Functional**

EDR is a technology designed and configured to secure endpoint devices that can detect and report unauthorised or anomalous phenomena in the system in a timely manner, and it is a technology used to detect violations of security policies in the network. Therefore the application of EDR systems requires the detection of intrusion attacks before they are directed at endpoint devices and the use of alarm and protection systems to expel intrusion attacks. During an intrusion attack, it is possible to reduce the damage caused by the intrusion attack.

In practice the trained model will be placed in the important network segment, monitoring the various packets in the segment constantly and characterising each packet or suspicious packet. If an abnormal endpoint is detected, the intrusion detection system can raise an alarm or even cut the network connection outright. In addition, after an intrusion attack, it is not only necessary to be able to identify the type of attack, but also to display detailed information about the source of the attack, such as IP address and port number.

**Non-functional**

After reviewing and analysing previous research papers, several ML algorithms are decided to be implemented in this project to detect malicious endpoints. Each algorithm will be individually realized and tested on both binary and multi-classification problems, and the ultimate goal is to combine them to form an architecture for detecting malicious endpoints. An integrated learning approach is also required when necessary, as individual learners are prone to under-fitting or over-fitting, and in order to obtain a learner with good generalisation performance, multiple individual learners can be trained to form a strong learner through some combination of strategies. Finally, multiple algorithms will be trained and tested on real IoT endpoint data to ensure the usability and effectiveness of models in a realistic environment.

It is crucial to select a valid dataset for research in this area. In most studies supervised learning based ML algorithms are proposed and therefore the chosen dataset should be labelled, containing normal, abnormal and various attack categories. Furthermore, there may be cases of null or abnormal values in the dataset, so methods such as data cleaning should be exploited to mitigate the influence of such data. In addition, the data redundancy and sample imbalance issues mentioned in section 2.4 are also significant. In order to solve such problems, the use of multiple feature selection and data sampling methods will effectively alleviate the imbalance problem and thus improve the classification performance. In the ML modelling process, various model evaluation metrics need to be used for different problems. Therefore, appropriate evaluation criteria such as Accuracy, Precision, Recall, etc. should be proposed before designing the model to demonstrate how well the model performs in detecting malicious endpoints. The time cost of training and testing the model should also be recorded.

It is also important to consider how to build models efficiently. The use of the Python programming language and ML libraries (scikit-learn, Keras, TensorFlow) can help in the implementation of algorithms. This is because these ML development frameworks offer a wide choice of algorithms and support data pre-processing and model evaluation operations. To maintain a high level of productivity and efficiency in projects from development to deployment and then in operations and maintenance.

**3.3 Data Selection Analysis**

It is vital to choose a valid dataset for this study. A series of recent studies have indicated that the most cited intrusion detection datasets, which are KDD-99 and NSL-KDD, have been gradually replaced. Due to evolving network traffic patterns, traditional datasets cannot be applied to current network endpoint devices, so it is critical to identify a comprehensive network dataset that reflects current traffic patterns. With the proliferation of IoT devices in recent years, the large amount of data generated by endpoint devices has driven the technology development in the field of intrusion detection. Therefore, this project will focus on IoT datasets to perform research on malicious endpoint detection and response.

**MQTT-IoT-IDS2020**

MQTT-IoT-IDS2020 is the first dataset to be created using the MQTT transport protocol. There were 12 sensors, a simulated camera and other devices to simulate an MQTT-based IoT network architecture. While capturing data from these sensors which communicate with each other via the MQTT protocol, and recording five scenarios including normal operation, aggressive scan, UDP scan, Sparta, and MQTT brute-force attack. The MQTT-IoT-IDS2020 dataset contains three abstraction levels of features are Uni-Flow, Bi-Flow and Packet-Flow respectively. And Figure 3 illustrates the distribution of five scenarios in three flow features.

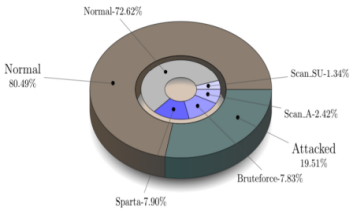
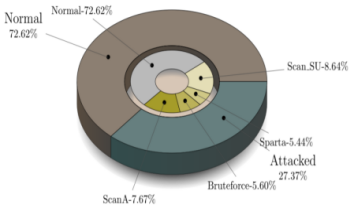
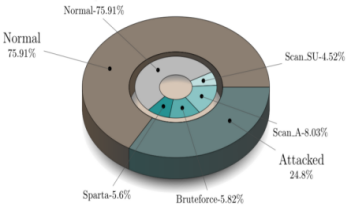


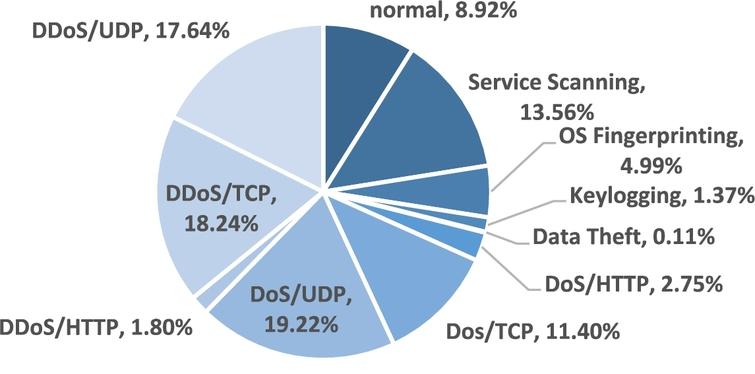
Figure 3.1Five files statistics of Uni-flow, Bi-flow and Packet-flow [52]

**MQTTset**

Similar to MQTT-IoT-IDS2020, the MQTTset dataset is composed of 8 different characteristics of MQTT sensors in a typical smart home, and the generated MQTT traffic is transformed into a PCAP file. The MQTTset dataset has six categories: Legitimate, Malformed, DoS attack, SlowITe, Bruteforce, and Flooding, so the dataset can reflect the real IoT environment.

**BoT-IoT**

The BoT-IoT dataset was built using Ubuntu VM to simulate an IoT network, and using the Node-red tool to simulate five IoT sensors to build a virtual IoT environment. And the MQTT protocol is used to transfer IoT messages to the cloud. The original dataset has 46 features and includes four categories respectively Normal, DoS, DDoS, Reconnaissance and Information Theft. However, the lack of monitoring of real IoT devices makes this dataset not really representative of IoT traffic [36] and it is clear from the Figure 3.2 that there is a significant sample imbalance problem in this dataset.



**Figure 3.2** The distribution of attack types of BoT-IoT dataset [36]

**IoT Network Intrusion (****IoT-NI)**

There were two IoT devices SKT NGU and EZVIZ Wi-Fi (C2C Mini O Plus 1080P) cameras used to create the IoT-NI dataset. The categories of the dataset are Normal, DoS, MITM, Mirai and Scan. With the exception of the Mirai botnet category, the other attack categories were constructed using packets collected when modelling network attacks using the Nmap software. However, it is clear from Figure 3.2 that the Mirai class dominates and the data sample is severely imbalanced.

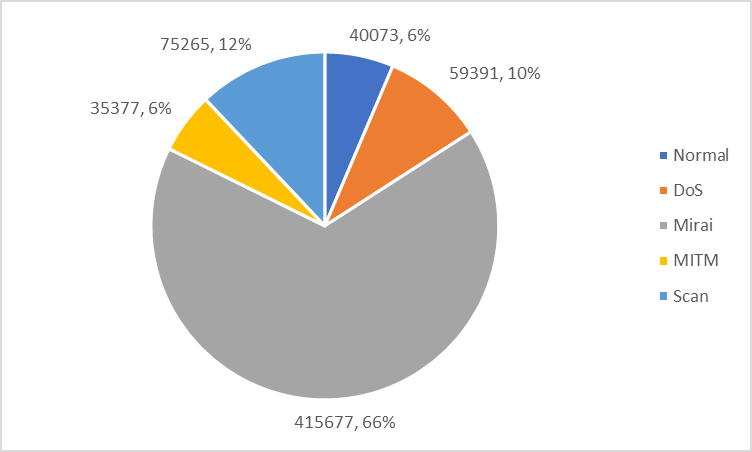


Figure 3.2 The distribution of attack types of IoT Network Intrusion dataset

**IoT-23**

Different from the BoT-IoT dataset, IoT-23's IoT traffic was collected from real hardware IoT devices. Amazon Echo devices, Philips Hue devices and Somfy door lock devices were used to capture this normal IoT packets. And malicious scenarios were created by executing malware using a Raspberry Pi again. There are 20 malicious scenarios and 3 benign scenarios in the IoT network included in the IoT-23 dataset. And Table 3.1 indicates the number of each category in the majority of labelled data.

|  |  |
| --- | --- |
| **Categories** | **Number** |
| C&C - File Download | 53 |
| C&C | 21995 |
| Benign | 30858735 |
| DDoS | 19538713 |
| C&C-Torii | 30 |
| File Download | 18 |
| Part-Of-A-Horizontal-PortScan | 213852924 |
| Attack | 9398 |
| C&C-Heart Beat | 33673 |
| C&C-Heart Beat File Download | 11 |
| C&C-Heart Beat Attack | 834 |
| C&C-Part-Of-A-Horizontal-PortScan | 888 |
| Okiru | 60990708 |
| Okiru-Attack | 3 |
| C&C-Mirai | 2 |
| Part-Of-A-Horizontal-PortScan-Attack | 5 |
| **Total** | 325307990 |

Table 3.1 The number of each category in the original dataset

**IoT-DS2**

Since an increasing number of IoT endpoints are targeted by malicious attacks. In order to tackle this challenge, a new dataset containing legitimate and malicious IoT network traffic, as well as various attack types need to be proposed. IoT-DS2 is a combination of five datasets - MQTT-IoT-IDS2020, MQTTset, BoT-IoT, IoT-NI, and IoT-23. The dataset captures complete network data containing a total of 83 features, 17 various scenarios were launched during the combination of the IoT-DS2 dataset. These scenarios are DDoS, Okiru, PortScan, Reconnaissance, Mirai, Sparta, MQQT\_bruteforce, Torii, C&C, DoS, Attack, Flood, HeartBeat, MITM ARP Spoofing, FileDownload, Theft, and Normal respectively. Moreover, this dataset includes 15 attack classes and one normal class that can be used to develop and accurately test malicious endpoint detection systems.

**3.4 Benchmark Datasets Comparison**

Valuable data contributes to training and evaluating the detection models, however there are a large number of datasets being proposed today for the detection of malicious attacks. Therefore, it is significant to review and compare the existing published datasets to find the best dataset representative for this project.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Year Published** | **Features**  **Numbers** | **Categories**  **Numbers** | **IoT**  **Purpose** | **MQTT Protocol** | **Realistic Traffic** | **Labeled** |
| KDD-99 | 1999 | 41 | 5 | ✕ | ✕ | ✕ | ✓ |
| NSL\_KDD | 2009 | 43 | 5 | ✕ | ✕ | ✕ | ✓ |
| UNSW-NB15 | 2015 | 49 | 9 | ✕ | ✕ | ✓ | ✓ |
| CICIDS2017 | 2017 | 85 | 15 | ✕ | ✕ | ✓ | ✓ |
| BoT-IoT | 2018 | 45 | 5 | ✓ | ✕ | ✓ | ✓ |
| TON\_IoT | 2019 | 45 | 10 | ✓ | ✓ | ✓ | ✓ |
| IoT-NI | 2019 | 42 | 5 | ✓ | ✓ | ✓ | ✓ |
| MQTT-IoT-IDS2020 | 2020 | 5 | 5 | ✓ | ✓ | ✓ | ✓ |
| MQTTset | 2020 | 33 | 6 | ✓ | ✓ | ✓ | ✓ |
| IoT-23 | 2020 | 21 | 10 | ✓ | ✕ | ✓ | ✓ |
| IoT-DS2 | 2021 | 83 | 17 | ✓ | ✓ | ✓ | ✓ |

Table 3.3 Summary of comparison for different benchmark datasets

There are 11 dominant datasets listed and compared in a number of respects in Table 3.3. Most of the recent research has been based on IoT network datasets, which use packet-based and flow-based as well as a combination of both to detect the attacks on IoT endpoints. However, traditional datasets such as KDD-99, NSL\_KDD, UNSW-NB15 and CICIDS2017 do not contain any characteristics of IoT applications and therefore do not effectively detect the malicious endpoints. In terms of the TON\_IoT dataset, contains heterogeneous data sources and has various normal and attack events for IoT endpoints[65], but it has some limitations due to the lack of authentication and disconnection phases. Furthermore, some studies with specific purposes such as BoT-IoT and IoT-23 do not address the raw traffic data of the MQTT protocol, but rather focus more on the DNS traffic data in the loT context.

The other type of dataset is the MQTT protocol-based dataset, which is widely used to communicate data between sensors or endpoints and this type of dataset best fits the IoT application scenario [66]. Therefore, it is crucial to use real traffic datasets that not only contain basic details about the endpoint device but also include the traffic that sensors are transmitting and clearly document the type of attack so that a comprehensive evaluation of the detection algorithms for malicious endpoints can be carried out. To tackle this problem Ullah, I. and Mahmoud, Q.H. proposed to combine five datasets to form the IoT-DS2 dataset, which can capture most of the attack types and remedy the shortcomings of other datasets to a certain extent. In addition, the dataset monitors the internal behaviours of IoT endpoints accurately, therefore it was selected as a benchmark dataset for evaluating detection models. But it suffers from data imbalance, therefore, the accuracy of detection for part of the attack types is not high.

**3.5** **Composition of IoT-DS2 Dataset**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Categories** | **MQTT-IoT-2020** | **MQTTSET** | **BoT-IoT** | **IoT-23** | **IoT-NI** | **IoT-DS2** |
| 1 | DDoS | - | - | 17420085 | - | - | 500000 |
| 2 | Normal | - | - | - | 4253672 | - | 2000000 |
| 3 | Okiru | - | - | - | 12908506 | - | 500000 |
| 4 | Port Scan | - | - | - | 2000000 | - | 500000 |
| 5 | Reconnaissance | - | - | 946268 | - | - | 500000 |
| 6 | Mirai | - | - | - | - | 366971 | 366971 |
| 7 | Sparta | 1217198 | - | - | - | - | 500000 |
| 8 | MQQT Bruteforce | 2001972 | - | - | - | - | 500000 |
| 9 | Torii | - | - | - | 24492 | - | 24492 |
| 10 | C&C | - | - | - | 20612 | - | 20612 |
| 11 | DoS | - | - | - | - | 59391 | 59391 |
| 12 | Attack | - | - | - | 1699608 | - | 500000 |
| 13 | Flood | - | 77756 | - | - | - | 77756 |
| 14 | HeartBeat | - | - | - | 12648 | - | 12648 |
| 15 | MITM ARP Spoofing | - | - | - | - | 32909 | 32909 |
| 16 | File Download | - | - | - | 7707 | - | 7707 |
| 17 | Theft | - | - | 445799 | - | - | 445799 |
| 18 | Malformed | - | 3535 | - | - | - | 3535 |
| 19 | SlowITe | - | 3044 | - | - | - | 3044 |
| **Total** | | | | | | | 6554864 |

Figure 3.4 The Composition of IoT-DS2 Dataset

Tables 3.4 indicates the composition information of the dataset named IoT-DS2. The latest IoT-DS2 dataset is composed of five IDS evaluation datasets, which are MQTT-IoT-IDS2020, MQTTset, BoT-IoT, IoT-NI, and IoT-23. It contains 18 attack categories and 1 normal category, but there is a significant imbalance in the samples, so the Malformed and SlowITe categories will be removed from the dataset. Moreover, The IoT-DS2 column represents the number of instances extracted from the five datasets, and also suffers from data redundancy, so with the weights of the classes being adjusted in the final dataset to divide the number of instances in each class. This dataset is available from [67] and can be efficiently used for the development and testing of a malicious IoT endpoint detection system. Furthermore, a series of data pre-processing steps, such as data cleaning, feature transformation, feature selection, data sampling and data segmentation, are still required before training and testing the model formally. In addtion, feature selection is a vital stage in constructing a deep learning model, not only to reduce the number of features to reduce training time, but also to make the model more generalisable and reduce over-fitting.

**Chapter 4**

**Design**

**4.1 System Architectural Design**

Figure 4.1 shows the overall framework of the system, from data processing to the creation of the ML model to the overall design of the actual prediction process. The first step in the process is to observe and analyse the collected data and perform data pre-processing operations. This involves data cleaning, data visualisation and feature replacement etc. After processing, the data should be split into training set, validation set and test set according to certain proportions. Feature engineering is then applied to the training set and data sampling techniques are used to address the sample imbalance problem. As well as the need to select features and perform normalization on the training set, validation set and test set before training the model. During the training of the model, the validation set is used to adjust the hyperparameters and select the best parameters, and it is also available to monitor the model for over-fitting during the training process. Finally, the proposed models are compared with other models and a comprehensive evaluation is carried out using different evaluation metrics.

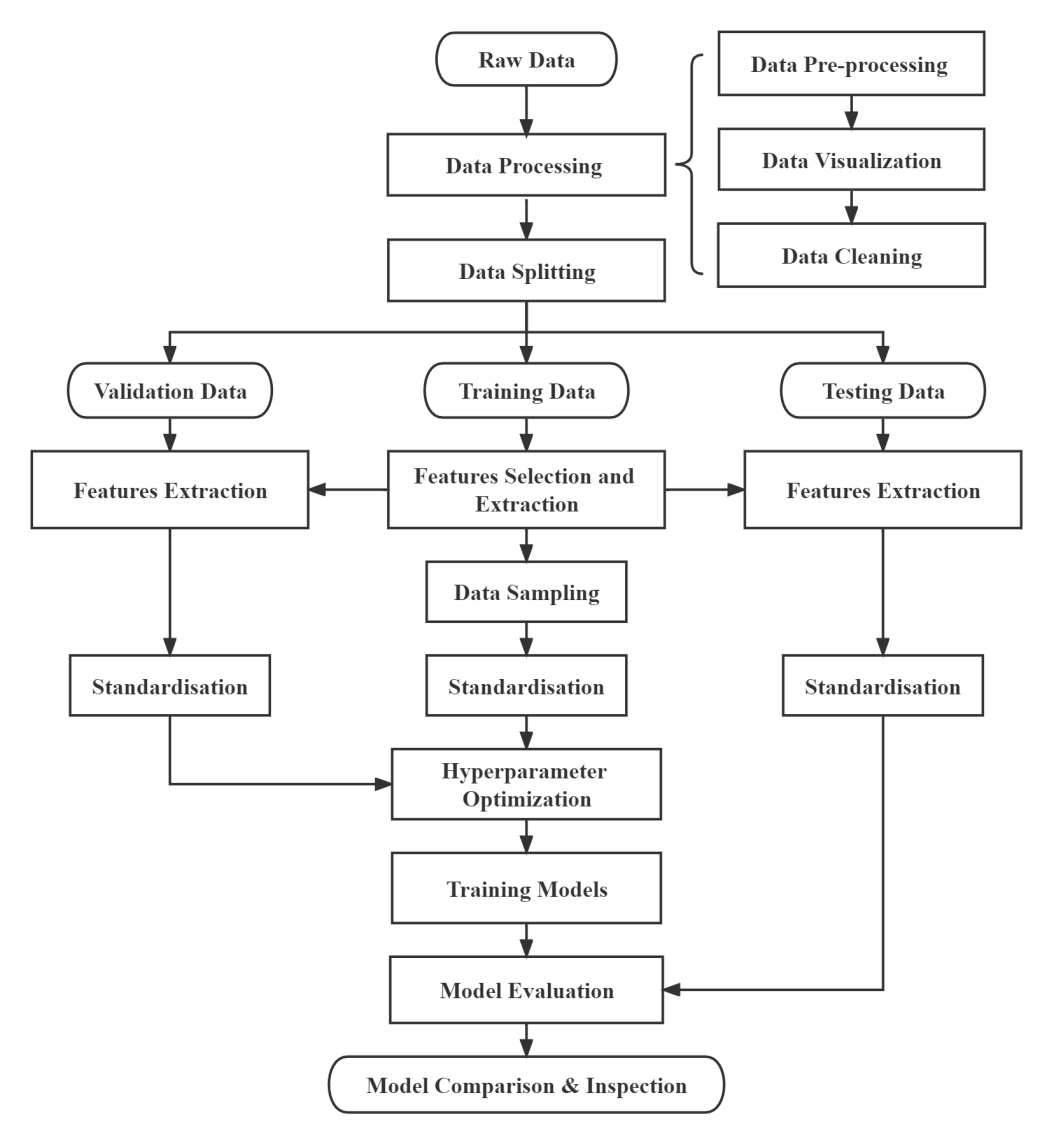
****

Figure 4.1 The structure of designing a malicious endpoint detection system

A practical application scenario of the EDR system is shown in Figure 4.2, which demonstrates the classification and prediction process of endpoint devices as it communicates and transmits data. The trained model should be deployed in each endpoint device to perform binary and multi-classification predictions for each data. In addition, it is important to respond promptly to statistics and output details of malicious endpoints and to store this information in a database to reduce the threat to sensitive data storage caused by malicious endpoints.

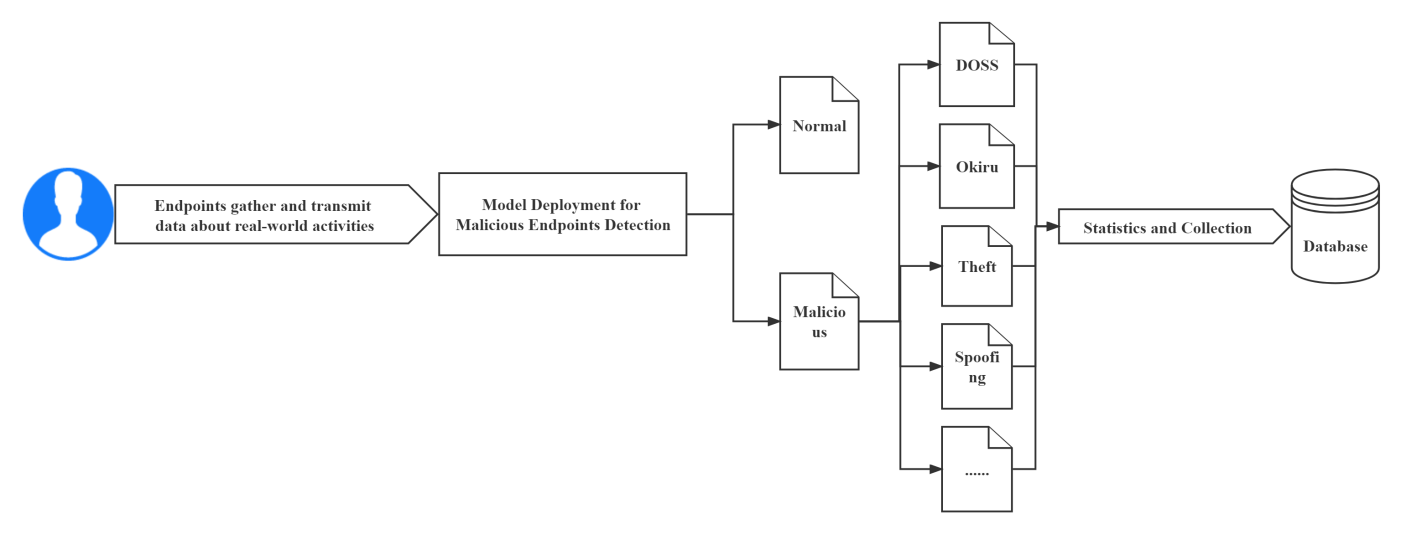
****

Figure 4.2 Real-time normal and malicious detection in endpoints

**4.2 Data Processing**

**4.2.1 Data Pro-processing and Cleaning**

|  |  |  |
| --- | --- | --- |
| **No** | **Features** | **Description** |
| 1 | Flow\_ID | Record the number of the transmitted data flow |
| 2 | Src\_IP | The source IP address of the host that is sending the packets |
| 3 | Src\_Port | The port number used by a program to send data to another program |
| 4 | Dst\_IP | The destination IP address of the host that is sending the packets |
| 5 | Dst\_Port | The port number used by a program to receive data from another program |
| 6 | Protocol | Textual representation of transaction protocols presents in network flow |
| 7 | Timestamp | Record the time that a traffic transaction occurred |
| 8 | Flow\_Duration | Record total duration |
| 9 | Pkts | Total count of packets in transaction |
| 10 | Max | Maximum duration of aggregated records |
| 11 | Min | Minimum duration of aggregated records |
| 12 | Mean | Average duration of aggregated records |
| 13 | Std | Standard deviation of aggregated records |
| 14 | Pkts/s | Count of packets in transaction per second |
| 15 | Byts/s | Count of bytes in transaction per second |
| 16 | Tot | Total duration of aggregated records |
| 18 | Var | Variance of aggregated records |
| 19 | Cnt | Flow state flags count seen in transactions |
| 20 | Ratio | Down or up ratio |
| 22 | Label | Class of label |
| 23 | Cat | Category of attack |

Table 4.1 The summary of utilized features

Data pre-processing is an efficient method of removing invalid data, irregular data and extremely erroneous data, so it is necessary to clean the data for a large dataset such as IoT-DS2. The IoT-DS2 dataset contains a total of 83 features, with 23 major feature types as shown in Figure 4.3. However, there are many missing and extreme values in the sample, for example, all of the features in 'Bwd\_IAT\_Mean.1' are missing values. If missing values in the data may reduce the statistical power of the research subjects or even lead to incorrect results due to biased estimation, which can have a significant impact on the modelling phase. Therefore, in order to eliminate anomalies in the original data, the first step is to find the very large, very small and missing values in the data set and interpolate them with the mean value. In terms of features with a large number of missing values, it is essential to count the percentage of missing values in each feature separately and set a threshold value and remove features that exceed that threshold.

**4.2.2 One-hot Encoding**

Because the dataset contains categorical data, the categorical features need to be coded from texttypes to numeric types in order to fit and evaluate the model before splitting the dataset into a test set and a training set. As shown in Figure 4.6 the 'Label' feature contains two categories 'Normal' and 'Anomaly', while the 'Cat' feature has a total of 17 different categories represented as text types. Therefore it is important to perform One-hot encoding of the target data using the OneHotEncoder class from the Scikit-Learn library, which is the process of converting the labelled data into numerical features of the dataset for use in the neural network. In addition, the mapping of discrete features to Euclidean space by one-hot encoding solves the problem of classifiers not handling attribute data well and serves to expand the features to some extent.

**4.2.3 Data Splitting**

### This section describes how to split the dataset in detail to form the training set, validation set and testing set. In a ML classification task, if a serious data imbalance problem is found in the samples, for example, the proportion of positive and negative samples is significantly different, it is necessary to perform stratified sampling. In the IoT-DS2 dataset, there is a serious sample imbalance problem in multiple classification tasks, so the simple random sampling method of splitting the dataset is no longer applicable, and a stratified sampling method like ''StratifiedShuffleSplit'' method in the sklearn library is more preferred. It splits the dataset into training set and testing set in a stratified approach according to the specified class characteristics, ensuring that the proportion of samples in each class in the split is consistent with the original proportion in the overall dataset.



Figure 4.3 Splitting dataset into training set, validation set and testing set

During the actual splitting process, the dataset should be stratified into a training set (90%), test set (10%) at first, and then the training set is further divided into training data (81%) and validation data (9%) as shown in Figure 4.3. The advantage of dividing the dataset in this way is that it ensures more sample data to train the model and avoids the data crossover problem ensuring absolute isolation of the training, validation, and test sets, as well as the rationality of stratified sampling.

**4.2.4 Features Selection**

Feature selection is one of the very important steps in ML, which not only effectively reduces the model training time by eliminating redundant or irrelevant features but also contributes to improving the classification quality. There are several techniques available for feature extraction and selection, and in this study, there will be a comparison between two methods based on Pearson Correlation Coefficient (PCC) and random forest built-in feature importance. The data after feature selection will be tested in a RF model and select the better method.

A picture containing text, electronics, display

Description automatically generated

Figure 4.4 Splitting dataset into training set, validation set and testing set

# The PCC is used to detect the degree of linear correlation between two continuous variables and it ranges from [-1,1]. And the positive value indicates a positive correlation, the negative value means a negative correlation, and the larger the absolute value, the higher the degree of linear correlation. For example, the heat map of the correlation coefficient matrix between each feature in the IoT-DS2 dataset is shown in Figure 4.4. Thecorrelation coefficient can also be viewed as the ratio of the product of the covariance between two variables X and Y and the product of the standard deviations of the two variables. The equation of PCC can be evaluated as follows:

Similarly, tree-based feature importance has been widely used to select features. The principle of decision tree construction shows that the features closer to the top of the decision tree are able to classify the different categories of samples, which means that the feature closer to the root node is more important. In Figure 4.5, several decision tree models with different structures form a random forest, and the final output category will be decided by voting at the output of the model. In sklearn, an importance assessment of the features is performed by the decision tree based on the reduction of the Gini purity. For the features that divide each node in the decision tree, the feature importance is calculated as:

Importance impurity left\_impurity right\_impurity

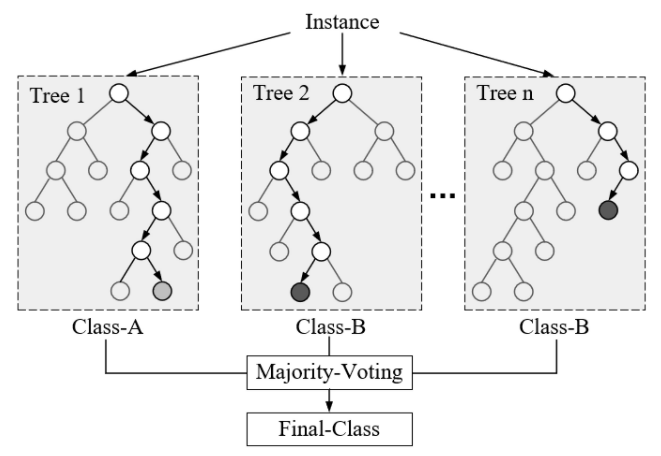


Figure 4.5 Random forest structure diagram

**4.2.5 Data Sampling**

The class imbalance problem is a common problem in reality. In most classification tasks, the number of data under each class is basically impossible to be exactly equal, but a small difference will not have any influence. In the case of some of the categories in the IoT-DS2 dataset, there is a serious data imbalance problem as shown in Figure 4.6, so further processing of the data in the training set is needed. The fundamental effect of sample imbalance is that the model learns a priori information about the proportion of samples in the training set so that the actual prediction will be biased towards most categories, which may lead to better accuracy in most categories and worse accuracy in a few categories, thereby affecting models’ robustness.

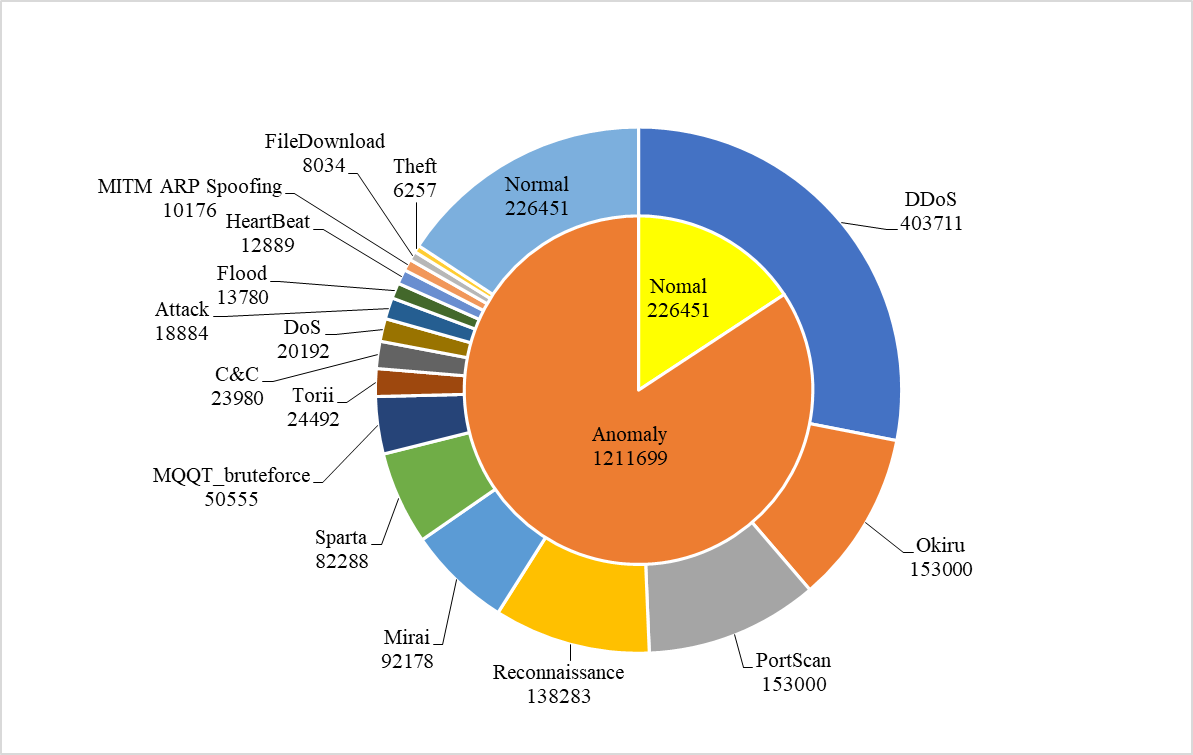


Figure 4.6 The number of different categories in IoT-DS2 dataset

Generally, the class imbalance problem can be mitigated at three levels, which are the sample level, the loss function level and the model level respectively. In terms of sample level, The undersampling algorithm adjusts the sample size by reducing the number of majority classes (e.g. random undersampling, NearMiss, ENN), while the oversampling algorithm increases the sample size of as many minority classes as possible (e.g. random oversampling, SMOTE) to achieve a balanced number of classes as shown in Figure 4.7.

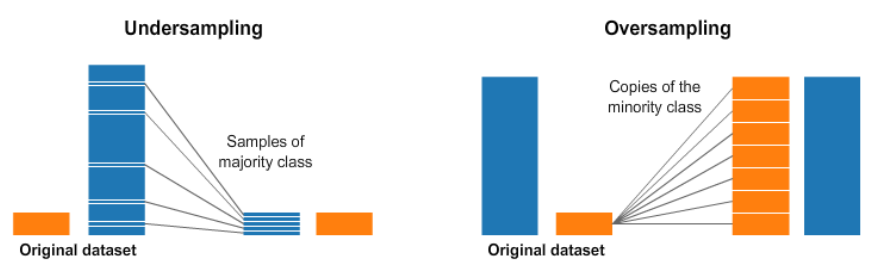
****

Figure 4.7 Oversampling and undersampling schematic diagram

Therefore it should be attempted to use multiple sampling methods to balance the sample size and compare it with the original samples. If this dataset cannot be handled effectively using sampling methods, the sample imbalance can be addressed by penalising weights for positive and negative samples. The categories in the classification are weighted high for small sample sizes and low for large sample sizes, which are calculated and modelled. As a result, the category balance is adjusted without increasing the computational complexity.

**4.2.6 Standardisation**

In terms of most of the ML algorithms and optimisation algorithms, scaling the feature values to the same range can lead to a better performing model. Generally, there are two common feature scaling algorithms, normalisation (Min-Max) and standardisation (Z-score), where the normalization algorithm scales the features to the range of [0,1] by the maximum and minimum values of the features. The general formula for a Min-Max of is given as:

However, in many ML algorithms, standardization may be better, which scales the features into a standard normal distribution with mean 0 and variance 1 by the mean and standard deviation of the features. Therefore, before training the model, the StandardScaler method in the sklearn library should be used to normalise the features by removing the mean and scaling to unit variance for the training, validation and testing sets respectively. The equation is shown as:

**4.3 Hyperparameter** **Optimization**

The hyperparameter optimisation plays a significant role in the performance of the model. In addition to manual parameter adjustment, there are two common hyperparameter optimisation techniques, respectively grid search and random search. Grid search optimization is the most fundamental method of hyperparameter optimisation arguably. With this technique it is simply a matter of constructing separate models for all possible hyperparameters, evaluating the performance of each model and selecting the model and hyperparameters that produce the best results. A disadvantage of grid search optimisation, however, is that the number of calculations grows exponentially when multiple hyperparameters are involved.

**4.4 Model Design**

In recent years, CNN and LSTM have been extensively investigated. CNNs have excelled in the fields of image recognition, natural language processing and speech recognition. In contrast, LSTM is the most successful in areas such as sentiment analysis and machine translation due to its pioneering structure of long and short-term memory. Therefore, in this study, a hybrid CNN-LSTM deep learning model is proposed to detect malicious endpoints.

**4.4.1 Convolutional Neural Network**

A classic CNN is comprised of four components as shown in Figure 4.8, namely the input layer, the convolutional layer, the pooling layer and the fully connected layer. The convolutional layer is the core of the algorithm and is responsible for extracting local features from the dat. The pooling layer is used to significantly reduce the size of the parameters (dimensionality reduction), and the fully connected layer is equivalent to a classifier and is used to output the desired result. Because the CNN is able to automatically perform feature extraction, share convolutional kernels to efficiently process high-dimensional data, as well as have high classification accuracy. The algorithm is suitable for the task of detecting malicious endpoints.

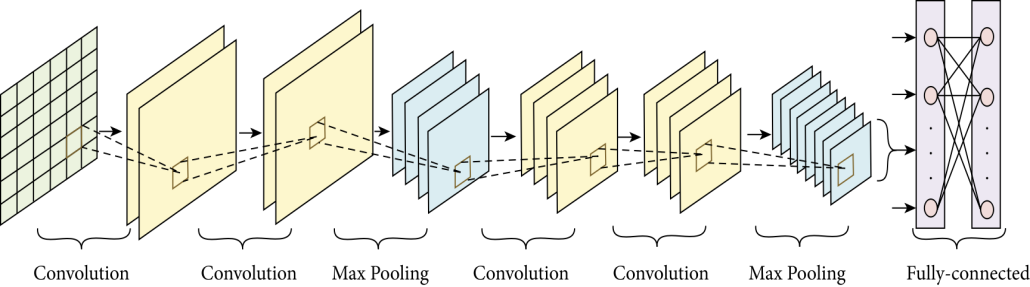


Figure 4.8 The structure of CNN algorithm

**4.4.2 Long Short Term Memory**

Long Short Term Memory is a special type of RNN designed to solve the problem of gradient disappearance and gradient explosion during the training of long sequences. LSTM differs from it by introducing three gates: the forgetting gate, the input gate, and the output gate. The forget gate determines which information should be discarded or kept. The LSTM requires an input gate to update the cell state. And the output gate determines the next hidden state, which is also used for prediction. Figure 4.9 demonstrates the structure of the LSTM model.

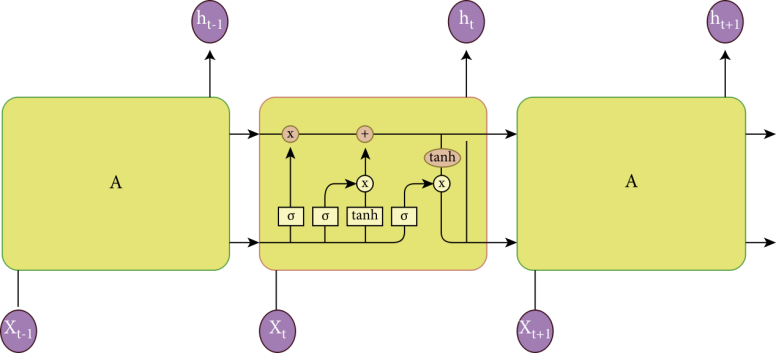


Figure 4.9 The structure of LSTM algorithm

**4.4.3 CNN-LSTM Algorithm**

Hybrid CNN and LSTM model is first proposed to detect and respond to malicious endpoints, and the generic structure of the hybrid CNN-LSTM model is shown in Figure 4.10.

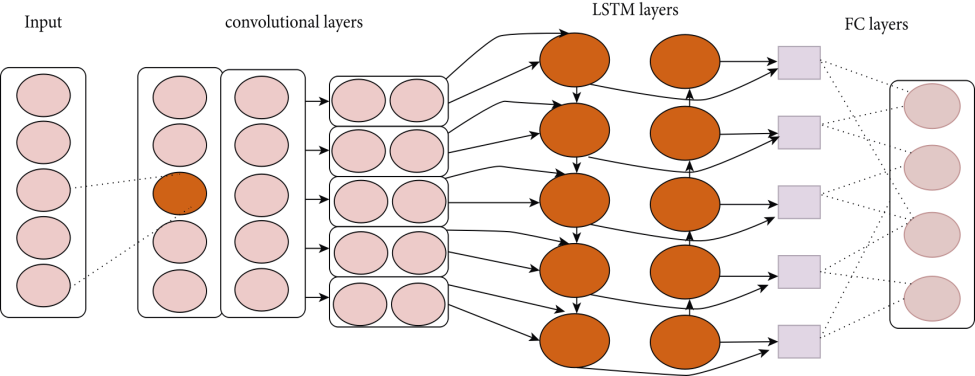


Figure 4.10 The structure of hybrid LSTM algorithm

In this project, it can be seen from Figure 4.14 that the proposed multi-classification model is composed of one input layer, seven convolutional layers, four pooling layers, two LSTM layers, two flatten layers and three fully connected layers.

Convolutional layers are used to reduce dimensions and extract features from the input data by convolutional operations. There are three significant hyperparameters in the convolutional layer, such as kernel size and activation function and padding, and the selection of the appropriate parameters has a positive effect in improving the performance of the convolutional layer.  It is similar to the fully-connected networks in which activation functions have to be added, such as sigmoid, tanh and ReLU. ReLu is a non-linear activation function. Compared to the sigmoid and tanh functions, ReLu will make the output of some of the neurons zero, which causes sparsity in the network and reduces the interdependence of the parameters, thus alleviating the overfitting problem.

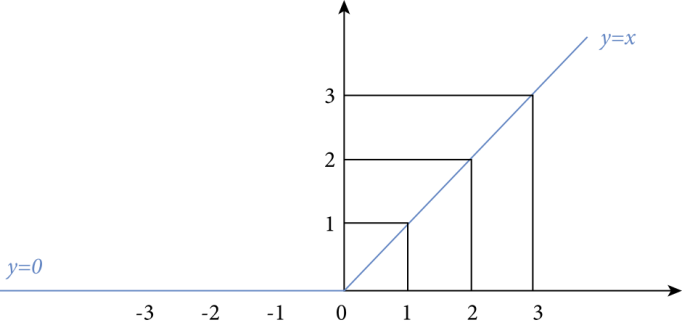


Figure 4.11 ReLu activation function overview diagram

The ReLU activation function is shown in the Figure 4.11 and the expression is as follows:

And the pooling layer, also known as the subsampling layer. It is an important concept in convolutional neural networks and is actually a form of downsampling, as shown in Figure 4.12. There are several different forms of non-linear pooling functions, of which 'Max pooling' is the most common. It involves dividing the input data into a number of rectangular regions and outputting the maximum value for each subregion. The pooling layer continually reduces the spatial size of the data and the number of parameters and the amount of computation, which also controls overfitting to some extent. In general, pooling layers are periodically inserted between the convolutional layers of a CNN.

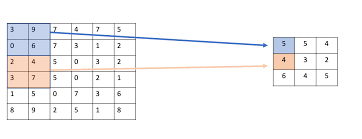


Figure 4.12 The pooling layeroverview diagram

Moreover, the fully-connected layer plays the role of a "classifier" in the entire convolutional neural network. Multiple fully-connected layers can be used in the same network, as shown in Figure 4.13, which increases the number of neurons, which increases the complexity of the model and thus improves its non-linear representation. However, it may lead to additional computational effort or over-fitting of the model.

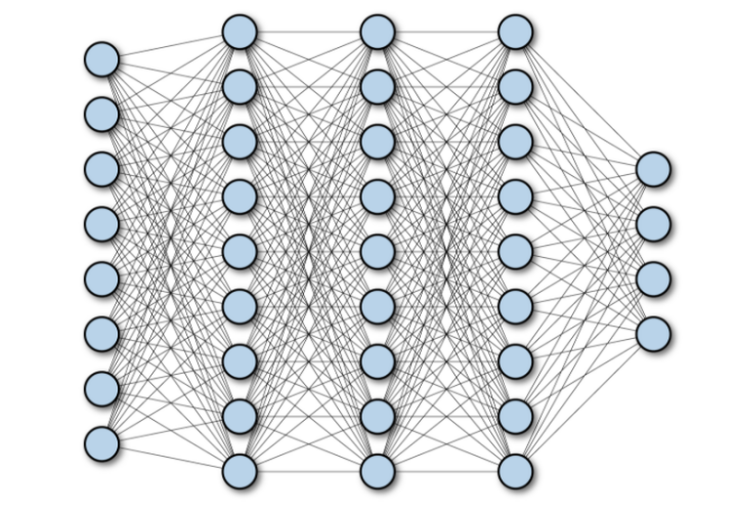


Figure 4.13 The fully-connected layer overview diagram

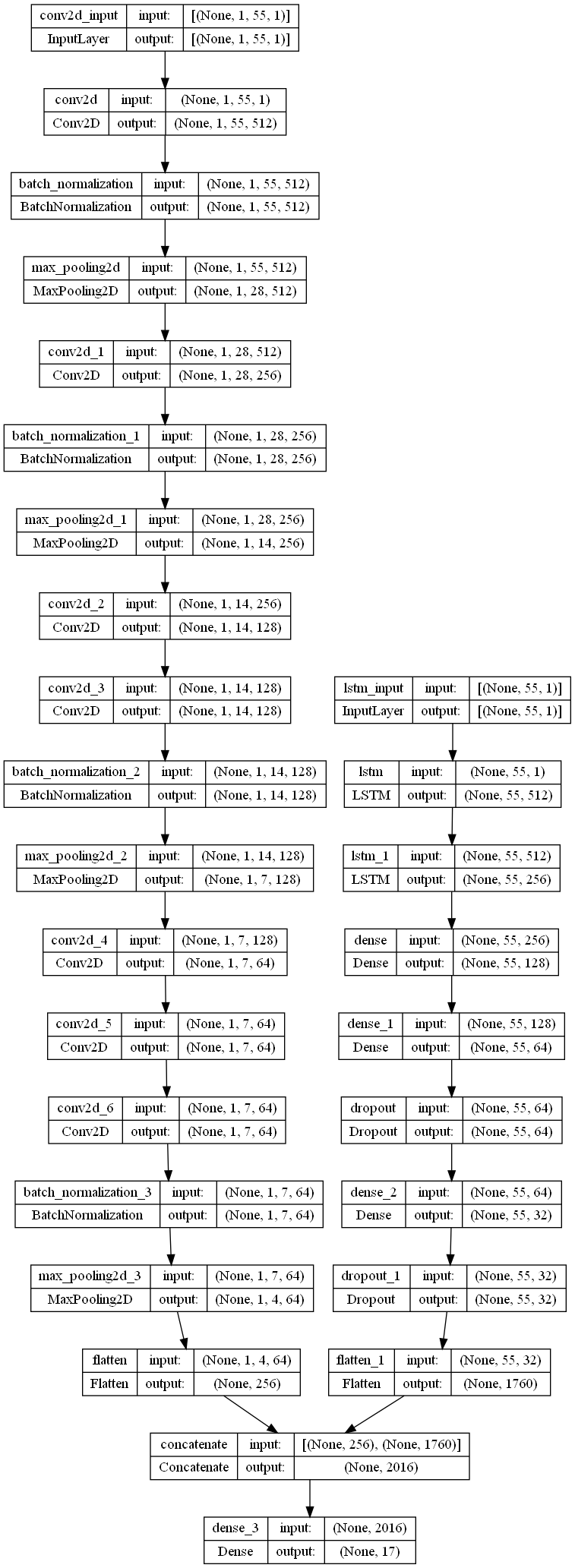


Figure 4.14 Proposed CNN-LSTM model architecture in detail

**4.5 Evaluation Metrics**

It is important to select the correct metrics for evaluating a model to effectively test how well the model performs on the test set. In general, evaluating models can be measured by a number of methods, as different metrics depending on the actual requirements of the particular research subject, and some models may perform well under one metric, but may not perform well under other metrics. Consequently, the binary and multi-classification models are evaluated in this study using confusion matrices, accuracy, precision, recall, F1-scores and other approaches.

**Confusion matrix**

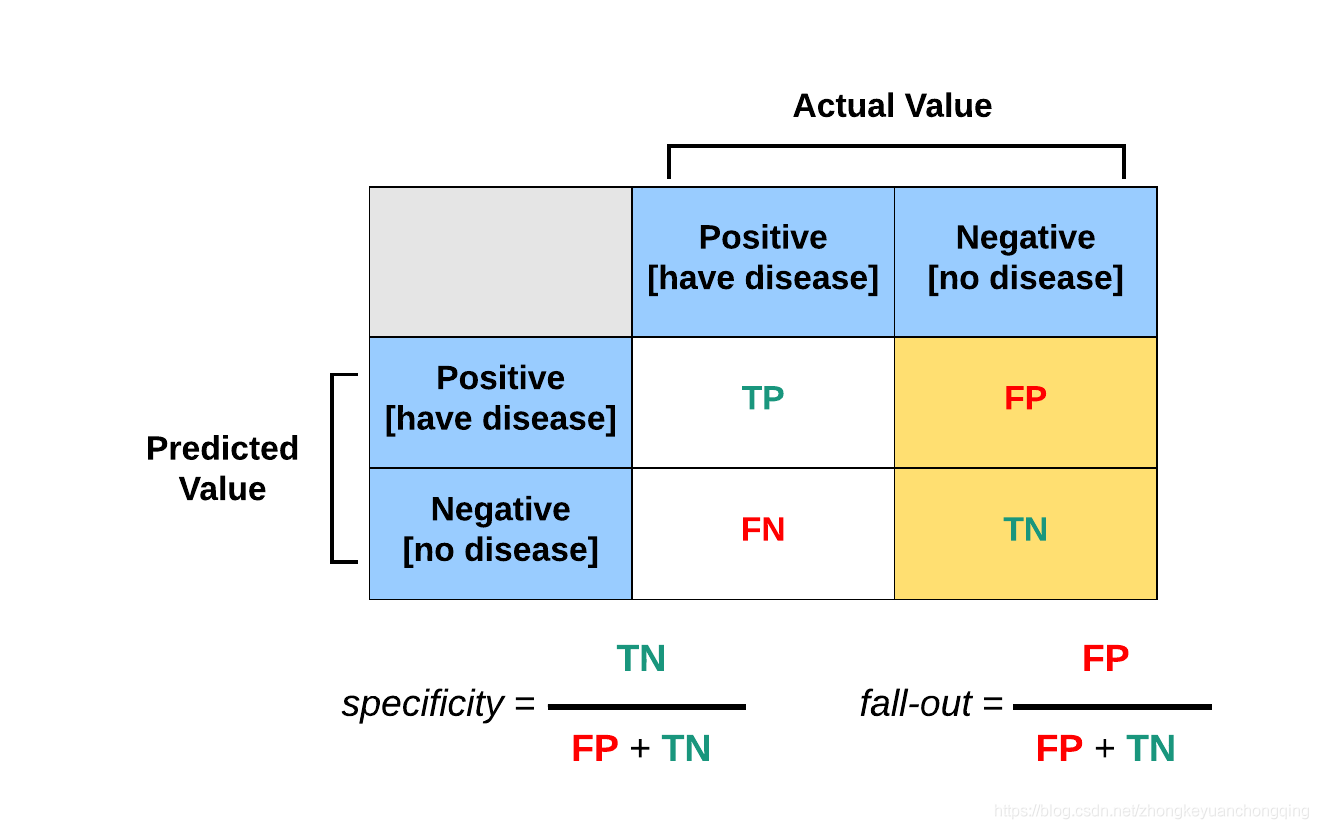


Figure 4.15 Confusion Matrix Example

A confusion matrix as shown in Figure 4.15 is a table of scenario analysis in ML that summarises the predicted results of a classification model, summarising the data in the form of a matrix of records in a dataset according to the true category and the predicted category. Each column of the confusion matrix represents the predicted category, and the total number of columns represents the number of data predicted to be in that category. Each row represents the true category to which the data belongs, and the total number of data in each row represents the number of instances of data in that category.

TP: True Positive Class. The true class of the sample is positive and the model identifies a positive class.

FN: False Negative Class. The true class of the sample is positive, but the model identifies it as negative.

FP: False Positive Class. The true class of the sample is negative, but the model identifies it as positive.

TN: True Negative Class. The true class of the sample is negative and the model identifies it as negative.

**Accuracy**

Accuracy is defined as the proportion of all samples that are predicted accurately. However, it has the disadvantage that it does not apply to unbalanced data.The formula is as follows:

**Precision**

Precision is the percentage of positive cases predicted correctly to the total number of cases predicted to be positive. It shows the ability of the model to detect anomalies and can also be referred to as the detection rate. In general, the higher the precision rate, the better the model is. The formula is as follows:

**Recall**

Recall refers to the ratio of the number of positive prediction matches to all positive cases, as know as sensitivity. This is one of the most important metrics for evaluating a model. It is because if a MEDR system does not detect an attack instance and predicts a malicious endpoint as a normal endpoint, a serious security problem can emerge. Usually, the higher the Recall, the more positive class samples are correctly predicted by the model and the more effective the model is. And the formula is as follows:

**F1-score**

F1-score is a measure score calculated by combining the results of precision and recall. When a model has a high precision return but a low recall score, the f-measure prefers to find a result by finding an average of precision and recall. The formula is as follows:

**TNR**

True negative rate is the proportion of true negative samples out of actual negative samples, also known as specificity. In general, the higher the true negative rate, the more effective the model is. The formula is as follows:

**FPR**

False positive rate: This indicates the number of negative samples predicted to be positive cases, as a proportion of the number of samples in the negative class of the model. In general, the lower the False positive rate, the better the model is. The formula is as follows:

**NPV**

The negative predictive value is the proportion of samples in which the model predicts a negative class and the proportion of samples that are actually negative. In general, the higher the NPV, the better the model is. The formula is as follows.

**FNR**

The false negative rate is the ratio of the number of samples in which the model predicts a negative class to the number of samples in which the class is positive to the true positive class. The smaller the missing value, the more effective the model is. The formula is as follows:

**FDR**

The false discovery rate is the proportion of samples with a true negative class among those predicted by the model to be positive. In general, the smaller the FDR, the better the model is. The formula is as follows:

**ROC**

The ROC graph is a curve that describes the relationship between sensitivity and specificity. The horizontal X-axis is the specificity, also known as the false positive rate, and the closer the X-axis is to zero, the better the accuracy. The vertical Y-axis is the sensitivity, also known as the true positive rate (sensitivity), and the larger the Y-axis, the better the accuracy. The area under the curve is called the AUC (Area Under Curve) and is used to indicate predictive accuracy. The higher the AUC value, the larger the area under the curve, and the higher the predictive accuracy.

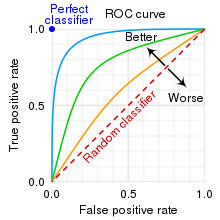


Figure 4.16 Confusion Matrix Example

**Chapter 5**

**Implementation and Testing**

This section describes the tools and techniques used in the project in detail. The research project is being completed on Jupyter Notebook using the Python programming language. A number of python data analysis libraries such as Pandas and NumPy will be used in the data pre-processing stage. In addition, ML frameworks such as scikit-learn, Keras and TensorFlow are also used to build ML models. These libraries need to be downloaded and installed in accordance with Table 5.1. The IoT-DS2 dataset is used to build a model for detecting malicious endpoints, with the initial step being to import the data for data pre-processing operations.

|  |  |
| --- | --- |
| **Name** | **Version** |
| Python | 3.10.1 |
| NumPy | 1.22.3 |
| Pandas | 1.4.3 |
| Sklearn | 1.1.1 |
| Keras | 2.9.0 |
| TensorFlow | 2.9.01 |

Table 5.1The version of libraries and tools

**5.1 Data Processing**

**5.1.1 Data Cleaning**

After importing the IoT-DS2 dataset, it is necessary to perform a data cleaning procedure. Because there are a significant number of abnormal values in the data, such as missing, infinite, and infinitesimal values, which cannot be processed in the model, therefore, it is necessary to delete or replace them. Firstly finding the infinitesimal and infinite values and replacing them with NaN, then removing any rows where there are extreme values. Then, all the 0 values in the dataset are replaced with NaN, and calculate the percentage of NaN in each feature to filter out the invalid features by removing any features with more than 90% of 0 values. In addition, it can be seen in Table. 5.3 that the number and percentage of NaN (0 values) in each feature. After the data cleaning operation, the dimensional change of the IoT-DS2 dataset is shown in Table. 5.2. As a result of data cleansing, some abnormal or missing data records in the dataset are replaced or deleted, and this procedure reduces the data dimensionality effectively.

|  |  |  |  |
| --- | --- | --- | --- |
| **Before Processing** | | **After Processing** | |
| Number of Features | Number of Rows | Number of Features | Number of Rows |
| 86 | 1438157 | 69 | 1438150 |

Table 5.2 The comparison before and after data cleaning for IoT-DS2 datasets

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Features** | **Number** | **[Percentage](javascript:;)** | **No** | **Features** | **Number** | **[Percentage](javascript:;)** |
| 1 | Flow\_ID | 0 | 0% | 44 | Fwd\_Pkt\_Len\_Max | 878708 | 61.1% |
| 2 | Label | 0 | 0% | 45 | Fwd\_Act\_Data\_Pkts | 884314 | 61.49% |
| 3 | Flow\_IAT\_Max | 0 | 0% | 46 | Fwd\_Act\_Data\_Pkts | 887813 | 61.733% |
| 4 | Flow\_IAT\_Mean | 0 | 0% | 47 | TotLen\_Bwd\_Pkts | 887813 | 61.733% |
| 5 | Flow\_Pkts/s | 0 | 0% | 48 | Bwd\_Pkt\_Len\_Mean | 887813 | 61.733% |
| 6 | Flow\_Duration | 0 | 0% | 49 | Bwd\_Pkt\_Len\_Max | 887813 | 61.733% |
| 7 | Cat | 0 | 0% | 50 | Subflow\_Bwd\_Byts | 896173 | 62.314% |
| 8 | Timestamp | 0 | 0% | 51 | Subflow\_Fwd\_Byts | 929092 | 64.603% |
| 9 | Src\_IP | 0 | 0% | 52 | Bwd\_IAT\_Tot | 941007 | 65.432% |
| 10 | Dst\_IP | 0 | 0% | 53 | Bwd\_IAT\_Mean | 941007 | 65.432% |
| 11 | Protocol | 2585 | 0.18% | 54 | Bwd\_IAT\_Max | 941007 | 65.432% |
| 12 | Dst\_Port | 2585 | 0.18% | 55 | Bwd\_IAT\_Min | 941632 | 65.475% |
| 13 | Src\_Port | 2593 | 0.18% | 56 | Pkt\_Len\_Std | 1032759 | 71.812% |
| 14 | Flow\_IAT\_Min | 10993 | 0.764% | 57 | Pkt\_Len\_Var | 1032759 | 71.812% |
| 15 | Init\_Fwd\_Win\_Byts | 12798 | 0.89% | 58 | Fwd\_Pkt\_Len\_Std | 1090160 | 75.803% |
| 16 | Tot\_Fwd\_Pkts | 117920 | 8.199% | 59 | Bwd\_Pkt\_Len\_Std | 1094023 | 76.072% |
| 17 | Fwd\_Pkts/s | 117921 | 8.199% | 60 | ACK\_Flag\_Cnt | 1177392 | 81.869% |
| 18 | Fwd\_Header\_Len | 120488 | 8.378% | 61 | Bwd\_IAT\_Std | 1181747 | 82.171% |
| 19 | Fwd\_IAT\_Max | 189260 | 13.16% | 62 | Fwd\_Pkt\_Len\_Min | 1219918 | 84.826% |
| 20 | Fwd\_IAT\_Tot | 189260 | 13.16% | 63 | Bwd\_Pkt\_Len\_Min | 1221065 | 84.905% |
| 21 | Fwd\_IAT\_Mean | 189260 | 13.16% | 64 | Pkt\_Len\_Min | 1225617 | 85.222% |
| 22 | Fwd\_IAT\_Min | 198817 | 13.824% | 65 | Down/Up\_Ratio | 1243138 | 86.44% |
| 23 | SYN\_Flag\_Cnt | 282818 | 19.665% | 66 | Active\_Max | 1246124 | 86.648% |
| 24 | Subflow\_Fwd\_Pkts | 291717 | 20.284% | 67 | Active\_Min | 1246124 | 86.648% |
| 25 | Flow\_IAT\_Std | 517917 | 36.013% | 68 | Active\_Mean | 1246124 | 86.648% |
| 26 | Idle\_Max | 585825 | 40.735% | 69 | PSH\_Flag\_Cnt | 1289612 | 89.672% |
| 27 | Idle\_Max | 585825 | 40.735% | 70 | FIN\_Flag\_Cnt | 1298300 | 90.276% |
| 28 | Idle\_Min | 585825 | 40.735% | 71 | Bwd\_Pkts/b\_Avg | 1358895 | 94.489% |
| 29 | Tot\_Bwd\_Pkts | 599813 | 41.707% | 72 | Bwd\_Blk\_Rate\_Avg | 1358895 | 94.489% |
| 30 | Bwd\_Pkts/s | 599814 | 41.707% | 73 | Active\_Std | 1403946 | 97.622% |
| 31 | Bwd\_Header\_Len | 600656 | 41.766% | 74 | RST\_Flag\_Cnt | 1408254 | 97.921% |
| 32 | Fwd\_IAT\_Std | 662418 | 46.06% | 75 | Fwd\_PSH\_Flags | 1429429 | 99.394% |
| 33 | Fwd\_Seg\_Size\_Min | 666282 | 46.329% | 76 | Bwd\_PSH\_Flags | 1432345 | 99.596% |
| 34 | Init\_Bwd\_Win\_Byts | 694442 | 48.287% | 77 | Bwd\_URG\_Flags | 1437698 | 99.969% |
| 35 | Idle\_Std | 712613 | 49.551% | 78 | URG\_Flag\_Cnt | 1437698 | 99.969% |
| 36 | Subflow\_Bwd\_Pkts | 773610 | 53.792% | 79 | CWE\_Flag\_Count | 1437917 | 99.984% |
| 37 | Pkt\_Size\_Avg | 841523 | 58.514% | 80 | ECE\_Flag\_Cnt | 1437949 | 99.986% |
| 38 | Pkt\_Len\_Mean | 841523 | 58.514% | 81 | Fwd\_Blk\_Rate\_Avg | 1438150 | 100% |
| 39 | Pkt\_Len\_Max | 841523 | 58.514% | 82 | Fwd\_Pkts/b\_Avg | 1438150 | 100% |
| 40 | Flow\_Byts/s | 841523 | 58.514% | 83 | Fwd\_URG\_Flags | 1438150 | 100% |
| 41 | Fwd\_Seg\_Size\_Avg | 878708 | 61.1% | 84 | Bwd\_Byts/b\_Avg | 1438150 | 100% |
| 42 | Fwd\_Pkt\_Len\_Mean | 878708 | 61.1% | 85 | Fwd\_Byts/b\_Avg | 1438150 | 100% |
| 43 | TotLen\_Fwd\_Pkts | 878708 | 61.1% |  |  |  |  |

Table 5.3 The number and percentage of 0 values (NaN) in each feature

**5.1.2 One-Hot** **Encoding**

The IoT-DS2 is a multi-category task discrete feature dataset, so that not only detects whether the security attributes of endpoints are normal or abnormal in the processing of classification tasks, but also allows multi-category classification of abnormal endpoint devices in detail. The data in the two predicted labels 'Label' and 'Cat' are of string type and need to be encoded, therefore label encoding is performed on the binary classification labels and One-Hot Encoding is performed on the multi-category labels.

Firstly, the binary and muti-class labels are encoded separately, replacing the string type with the corresponding numeric type. Label encoding is the process of assigning a numerical label to a feature value in the form of a string based on its position in the feature sequence, which is used to provide a learning model based on numerical algorithms as shown in Table 5.4 .

However, numerical values can be misunderstood by the algorithm as having some sort of hierarchical structure or order, so label encoding is obviously limited. This is because in ML algorithms such as regression, classification, and clustering, the calculation of distance or similarity between features is very important, and the common calculation of distance or similarity is in the Euclidean space of similarity.

In terms of multi-class tasks, therefore, it is significant to process One-hot encoding before the training model. After One-hot encoding, the dimensionality of the multi-categorical labels has changed from 1×1164901 to 17×1164901.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Binary** **Classification** | | | **Muti-class Classification** | | |
| Categories | Number Encoding | Amount | Categories | Number Encoding | Amount |
| Normal | 0 | 226451 | Normal | 0 | 226451 |
| Abnormal | 1 | 1211699 | DDoS | 1 | 403711 |
| PortScan | 2 | 153000 |
| Okiru | 3 | 153000 |
| Reconnaissance | 4 | 138283 |
| Mirai | 5 | 92178 |
| Sparta | 6 | 82288 |
| MQQT\_bruteforce | 7 | 50555 |
| Torii | 8 | 24492 |
| C&C | 9 | 23980 |
| DoS | 10 | 20192 |
| Attack | 11 | 18884 |
| Flood | 12 | 13780 |
| HeartBeat | 13 | 12889 |
| MITM ARP Spoofing | 14 | 10176 |
| FileDownload | 15 | 8034 |
| Theft | 16 | 6257 |
| Total | |  | | | 1438150 |

Table 5.4 Label encoding for binary and muti-class

**5.1.3 Data Splitting**

In order to split the training, validation and testing sets, the stratified sampling method of the ''StratifiedShuffleSplit'' function was used to perform this task. As can be seen from Table 5.5, the dataset was finally split into training, validation and testing set according to a defined proportion, and the problem of data duplication and crossover was avoided successfully.

|  |  |  |
| --- | --- | --- |
| **Datasets** | **Size (Rows)** | **Percentage** |
| Training Set | 1164901 | 81% |
| Validation Set | 129434 | 9% |
| Testing Set | 143815 | 10% |
| Total | 1438150 | |

Table 5.5 Splitting dataset into training, validation and testing sets in detailed

**5.1.4 Features Selection**

Before selecting features using Pearson's correlation coefficient or random forest built-in feature importance, some features that are not relevant to the classification result are removed. For example, "Flow\_ID", "Src\_IP", "Src\_Port", "Dst\_IP", "Dst\_Port", "Protocol", and "Timestamp", which are details of the endpoint devices in the flow data, are not relevant when training the classification model. However, when using the model for detection, these details of malicious endpoints need to be output to enable real-time response to malicious endpoints. In the binary and multi-classification tasks, Pearson's correlation coefficient and random forest built-in feature importance were used to select the appropriate features respectively. A reasonable threshold was set to filter out features of low importance.

It can be seen from Table 5.6 and Table 5.8 that the correlation coefficient between each feature and the categorical variable is calculated, and a random forest model is trained with the processed data. And using the "feature\_importances\_" parameter in sklearn to calculate the feature importance. It is clear from observation that the importance and correlation of features are not high with a large number of features. As a consequence, the thresholds were set to 0.001 and 0.03 for binary classification and 0.0014 and 0.06 for muti-class classification for the RF model importance method and the Pearson correlation coefficient method respectively.

Finally, the two feature selection methods were used to train multiple random forest models with the original data and test them separately. In terms of binary classification task, the test results of the three classification models are shown in Table 5.7. In comparison with the models trained on the original data, the feature-selected dataset showed a significant reduction in both model training and testing time. In comparison to the original data, the feature-selected dataset has a significant reduction in the number of features and in the time consumed to train and test the model. However, the Pearson correlation coefficient-based feature selection approach reduced the performance of the model, but the model importance-based approach reduced the computational complexity while keeping the classification accuracy largely unchanged. Furthermore, the accuracy of the test set was improved after feature selection using a feature importance-based approach in a multi-classification task. Therefore, the RF built-in feature importance method will be chosen to select features for the binary multi-class classification task, and the final number of features will be reduced from 67 to 33 and 55 respectively.

**Binary Classification**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Features** | **Importance** | **PCC** | **No** | **Features** | **Importance** | **PCC** |
| 1 | Fwd\_Seg\_Size\_Min | 0.12805 | -0.6456 | 31 | ACK\_Flag\_Cnt | 0.00149 | 0.11219 |
| 2 | SYN\_Flag\_Cnt | 0.09464 | -0.4314 | 32 | Pkt\_Len\_Min | 0.00146 | 0.08028 |
| 3 | Idle\_Mean | 0.07352 | -0.6559 | 33 | Init\_Bwd\_Win\_Byts | 0.00119 | 0.16870 |
| 4 | Init\_Fwd\_Win\_Byts | 0.07298 | -0.5160 | 34 | Fwd\_Pkt\_Len\_Max | 0.00096 | 0.03865 |
| 5 | Fwd\_IAT\_Min | 0.06499 | -0.0573 | 35 | Subflow\_Bwd\_Byts | 0.00071 | 0.04089 |
| 6 | Flow\_IAT\_Min | 0.05707 | 0.00033 | 36 | Fwd\_Seg\_Size\_Avg | 0.00067 | 0.09859 |
| 7 | Idle\_Max | 0.05092 | -0.6426 | 37 | TotLen\_Bwd\_Pkts | 0.00065 | 0.01735 |
| 8 | Fwd\_Header\_Len | 0.05031 | -0.0020 | 38 | Fwd\_Pkt\_Len\_Min | 0.00058 | 0.08269 |
| 9 | Idle\_Min | 0.04894 | -0.6906 | 39 | Fwd\_Pkt\_Len\_Mean | 0.00057 | 0.09859 |
| 10 | Bwd\_Pkts/s | 0.03758 | 0.07685 | 40 | Pkt\_Len\_Mean | 0.00055 | 0.04662 |
| 11 | Fwd\_IAT\_Std | 0.02824 | 0.05624 | 41 | Pkt\_Size\_Avg | 0.00044 | 0.05218 |
| 12 | Fwd\_IAT\_Max | 0.02622 | 0.05957 | 42 | Bwd\_IAT\_Max | 0.00035 | 0.11816 |
| 13 | Flow\_IAT\_Mean | 0.02221 | -0.0102 | 43 | Bwd\_Pkt\_Len\_Mean | 0.00035 | 0.02951 |
| 14 | Flow\_IAT\_Max | 0.02105 | 0.07497 | 44 | TotLen\_Fwd\_Pkts | 0.00032 | 0.04700 |
| 15 | Tot\_Fwd\_Pkts | 0.02078 | 0.00114 | 45 | Pkt\_Len\_Var | 0.00031 | 0.01150 |
| 16 | Tot\_Bwd\_Pkts | 0.02070 | 0.18158 | 46 | Subflow\_Fwd\_Byts | 0.00027 | 0.10684 |
| 17 | Fwd\_IAT\_Tot | 0.01986 | 0.03415 | 47 | Fwd\_Act\_Data\_Pkts | 0.00026 | 0.16195 |
| 18 | Fwd\_IAT\_Mean | 0.01926 | -0.0060 | 48 | Pkt\_Len\_Std | 0.00021 | 0.03242 |
| 19 | Bwd\_Header\_Len | 0.01794 | 0.15783 | 49 | Bwd\_Pkt\_Len\_Min | 0.00020 | 0.0370 |
| 20 | Idle\_Std | 0.01541 | 0.06566 | 50 | Bwd\_IAT\_Mean | 0.00017 | 0.13545 |
| 21 | Flow\_IAT\_Std | 0.01495 | 0.04698 | 51 | Bwd\_IAT\_Tot | 0.00017 | 0.12251 |
| 22 | Subflow\_Bwd\_Pkts | 0.01383 | 0.26622 | 52 | Bwd\_Seg\_Size\_Avg | 0.00015 | 0.02951 |
| 23 | Fwd\_Pkts/s | 0.01266 | 0.20604 | 53 | PSH\_Flag\_Cnt | 0.00013 | 0.10154 |
| 24 | Flow\_Pkts/s | 0.01203 | 0.20812 | 54 | Fwd\_Pkt\_Len\_Std | 0.00009 | 0.03523 |
| 25 | Pkt\_Len\_Max | 0.00975 | 0.04663 | 55 | Down/Up\_Ratio | 0.00006 | 0.11538 |
| 26 | Bwd\_Pkt\_Len\_Max | 0.00933 | 0.03808 | 56 | Bwd\_IAT\_Std | 0.00003 | 0.09326 |
| 27 | Flow\_Byts/s | 0.00910 | 0.04887 | 57 | Bwd\_Pkt\_Len\_Std | 0.00001 | 0.03493 |
| 28 | Subflow\_Fwd\_Pkts | 0.00667 | 0.12412 | 58 | Active\_Min | 0.00000 | 0.08584 |
| 29 | Bwd\_IAT\_Min | 0.00439 | 0.10194 | 59 | Active\_Mean | 0.00000 | 0.09056 |
| 30 | Flow\_Duration | 0.00404 | 0.0763 | 60 | Active\_Max | 0.00000 | 0.09334 |

Table 5.6 Using feature importance and PCC on the binary classification to select features

**Compare the different predicted results for binary classification**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Metrics** | **Original Dataset** | | **Feature Importance** | | **PCC** | |
|  | Normal | Abnormal | Normal | Abnormal | Normal | Abnormal |
| Accuracy | 0.99550 | | 0.99542 | | 0.99540 | |
| Precision | 0.98 | 1.00 | 0.98 | 1.00 | 0.98 | 1.00 |
| Recall | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 | 1.00 |
| F1-score | 0.99 | 1.00 | 0.99 | 1.00 | 0.99 | 1.00 |
| Time | 1m 49s | | 1m 46s | | 1m 38s | |
| Number of features | 67 | | 33 | | 51 | |

Table 5.7 Compare the results using different feature selection methods for binary classification

**Muti-class Classification**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Features** | **Importance** | **PCC** | **No** | **Features** | **Importance** | **PCC** |
| 1 | Init\_Fwd\_Win\_Byts | 0.12924 | -0.1619 | 31 | Pkt\_Len\_Mean | 0.00502 | 0.19303 |
| 2 | Fwd\_Seg\_Size\_Min | 0.07289 | 0.07911 | 32 | Pkt\_Len\_Max | 0.00481 | 0.17946 |
| 3 | Bwd\_Pkts/s | 0.05400 | 0.16666 | 33 | Bwd\_Seg\_Size\_Avg | 0.00480 | 0.14311 |
| 4 | Flow\_IAT\_Max | 0.05130 | 0.11482 | 34 | Pkt\_Len\_Std | 0.00464 | 0.15320 |
| 5 | Idle\_Max | 0.04865 | 0.14525 | 35 | Tot\_Fwd\_Pkts | 0.00413 | 0.00595 |
| 6 | Subflow\_Bwd\_Pkts | 0.04854 | -0.0671 | 36 | Bwd\_IAT\_Max | 0.00366 | 0.11376 |
| 7 | SYN\_Flag\_Cnt | 0.04833 | -0.0624 | 37 | Bwd\_Pkt\_Len\_Mean | 0.00250 | 0.14311 |
| 8 | Flow\_IAT\_Mean | 0.04589 | -0.1623 | 38 | Idle\_Std | 0.00249 | 0.19459 |
| 9 | Idle\_Mean | 0.04197 | 0.13541 | 39 | Bwd\_IAT\_Std | 0.00244 | 0.13603 |
| 10 | Bwd\_Header\_Len | 0.03823 | 0.26074 | 40 | Pkt\_Size\_Avg | 0.00233 | 0.20097 |
| 11 | Flow\_Pkts/s | 0.03800 | -0.0663 | 41 | Fwd\_Act\_Data\_Pkts | 0.00229 | 0.08930 |
| 12 | Tot\_Bwd\_Pkts | 0.03177 | 0.25488 | 42 | Pkt\_Len\_Min | 0.00227 | 0.08908 |
| 13 | Fwd\_Pkts/s | 0.02952 | -0.0712 | 43 | Fwd\_Pkt\_Len\_Max | 0.00218 | 0.14465 |
| 14 | Idle\_Min | 0.02919 | 0.07415 | 44 | Subflow\_Fwd\_Byts | 0.00201 | 0.04184 |
| 15 | Flow\_Duration | 0.02715 | -0.0041 | 45 | Bwd\_Pkt\_Len\_Min | 0.00198 | 0.13353 |
| 16 | Fwd\_IAT\_Tot | 0.02660 | 0.06590 | 46 | Fwd\_Pkt\_Len\_Mean | 0.00197 | 0.21529 |
| 17 | Fwd\_Header\_Len | 0.02433 | 0.00777 | 47 | PSH\_Flag\_Cnt | 0.00182 | 0.23771 |
| 18 | Flow\_IAT\_Min | 0.02123 | -0.1322 | 48 | TotLen\_Fwd\_Pkts | 0.00179 | 0.18079 |
| 19 | ACK\_Flag\_Cnt | 0.02003 | 0.35285 | 49 | Fwd\_Seg\_Size\_Avg | 0.00176 | 0.21529 |
| 20 | Init\_Bwd\_Win\_Byts | 0.01787 | 0.03326 | 50 | Pkt\_Len\_Var | 0.00171 | 0.10321 |
| 21 | Fwd\_IAT\_Mean | 0.01654 | -0.0526 | 51 | Fwd\_Pkt\_Len\_Std | 0.00158 | 0.11065 |
| 22 | Fwd\_IAT\_Max | 0.01368 | 0.14375 | 52 | Subflow\_Bwd\_Byts | 0.00152 | 0.11415 |
| 23 | Flow\_IAT\_Std | 0.01034 | 0.10147 | 53 | Bwd\_Pkt\_Len\_Std | 0.00148 | 0.11758 |
| 24 | Bwd\_IAT\_Tot | 0.00980 | 0.10517 | 54 | TotLen\_Bwd\_Pkts | 0.00144 | 0.14323 |
| 25 | Fwd\_IAT\_Std | 0.00711 | 0.14917 | 55 | Bwd\_Pkt\_Len\_Max | 0.00141 | 0.14737 |
| 26 | Subflow\_Fwd\_Pkts | 0.00711 | -0.1485 | 56 | Fwd\_Pkt\_Len\_Min | 0.00129 | 0.21882 |
| 27 | Bwd\_IAT\_Mean | 0.00680 | -0.0567 | 57 | Down/Up\_Ratio | 0.00056 | 0.21236 |
| 28 | Fwd\_IAT\_Min | 0.00625 | -0.2375 | 58 | Active\_Min | 0.00003 | -0.1312 |
| 29 | Flow\_Byts/s | 0.00601 | 0.27774 | 59 | Active\_Max | 0.00001 | -0.1427 |
| 30 | Bwd\_IAT\_Min | 0.00543 | -0.1569 | 60 | Active\_Mean | 0.00001 | -0.1384 |

Table 5.8 Using feature importance and PCC on the muti-class classification to select features

**Compare the different predicted results for muti-class classification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metrics** | **Original Dataset** | **Feature Importance** | **PCC** |
| Avg accuracy | 0.98904 | 0.98913 | 0.98836 |
| Avg precision | 0.96 | 0.96 | 0.96 |
| Avg recall | 0.94 | 0.94 | 0.93 |
| Avg f1-score | 0.95 | 0.95 | 0.94 |
| Time | 2m 54s | 3m 1s | 2m 50s |
| Number of features | 67 | 55 | 53 |

Table 5.9 Compare the results using different feature selection methods for muti-class classification

**5.1.5 Data Sampling**

In order to address the problem of sample imbalance in the multi-classification task, some of the categories in training set were oversampled. By calculating the average of the total number of samples, a category with a number close to the average is selected as a benchmark and several other categories are oversampled using the SMOTE algorithm. A small number of category samples were analysed and simulated and new manually simulated samples were added to the dataset, resulting in the categories in the original data no longer being significantly imbalanced. As can be seen from Figure 5.1 and Figure 5.2 the categories with fewer samples have been sampled to a mean level. However, sometimes SMOTE improves classification accuracy and sometimes it does not, and even may lead to worse classification results due to the amplification of noisy data when constructing the data.

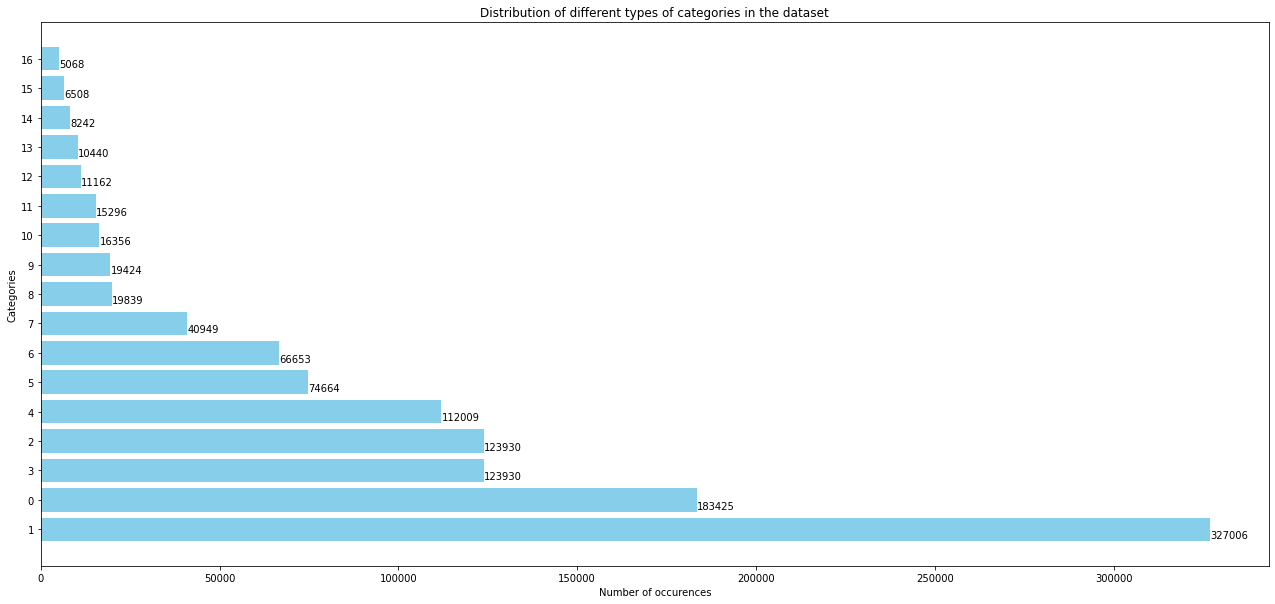


Figure 5.1 The number of each category before sampling

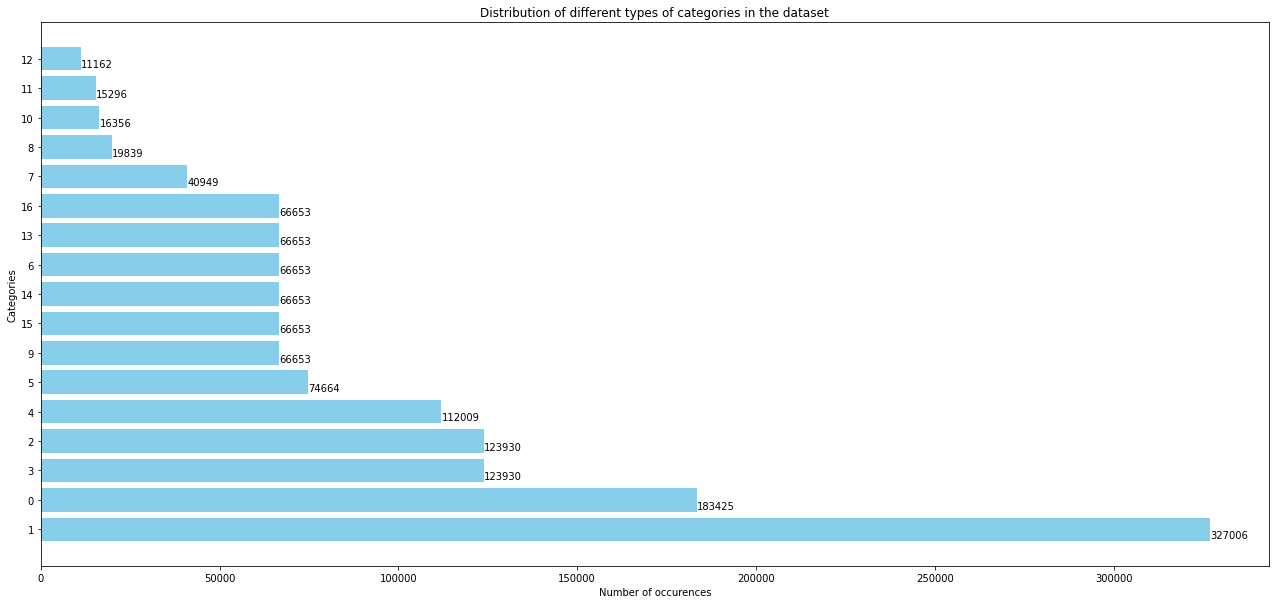


Figure 5.2 The number of each category after sampling

**5.2 Parameters Setting and Training**

The final CNN-LSTM model structure can be seen in Table 5.10, which shows in detail which layers are in the model, the order of each layer, the output shape as well as the number of parameters in the layers. After parameter selection, the setting parameters of the CNN-LSTM model are shown in Table 5.11. The convolutional kernel size in the convolutional layer is set to 3 × 3, the pooling kernel size is 2 × 2, and ReLU is used as the activation function to mitigate the gradient disappearance and make the network training faster. The model performed better when softmax and RMSprop were selected for the Classification function and Optimizer respectively. The number of Epochs was set to 20, in addition, EarlyStopping was set to monitor that the training was stopped when the change in loss value was less than 0.0001.

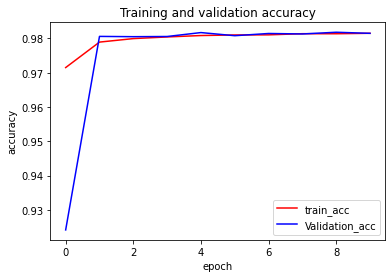
|  |  |  |
| --- | --- | --- |
| **Layer (type)** | **Output Shape** | **Param** |
| conv2d\_input (InputLayer) | [(None, 1, 55, 1)] | 0 |
| conv2d (Conv2D) | (None, 1, 55, 512) | 5120 |
| batch\_normalization (BatchNormalization) | (None, 1, 55, 512) | 2048 |
| max\_pooling2d (MaxPooling2D) | (None, 1, 28, 512) | 0 |
| conv2d\_1 (Conv2D) | (None, 1, 28, 256) | 1179904 |
| batch\_normalization\_1 (BatchNormalization) | (None, 1, 28, 256) | 1024 |
| max\_pooling2d\_1 (MaxPooling2D) | (None, 1, 14, 256) | 0 |
| conv2d\_2 (Conv2D) | (None, 1, 14, 128) | 295040 |
| conv2d\_3 (Conv2D) | (None, 1, 14, 128) | 147584 |
| lstm\_input (InputLayer) | [(None, 55, 1)] | 0 |
| batch\_normalization\_2 (BatchNormalization) | (None, 1, 14, 128) | 512 |
| lstm (LSTM) | (None, 55, 512) | 1052672 |
| max\_pooling2d\_2 (MaxPooling2D) | (None, 1, 7, 128) | 0 |
| lstm\_1 (LSTM) | (None, 55, 256) | 787456 |
| conv2d\_4 (Conv2D) | (None, 1, 7, 64) | 73792 |
| dense (Dense) | (None, 55, 128) | 32896 |
| conv2d\_5 (Conv2D) | (None, 1, 7, 64) | 36928 |
| dense\_1 (Dense) | (None, 55, 64) | 8256 |
| conv2d\_6 (Conv2D) | (None, 1, 7, 64) | 36928 |
| dropout (Dropout) | (None, 55, 64) | 0 |
| batch\_normalization\_3 (BatchNormalization) | (None, 1, 7, 64) | 256 |
| dense\_2 (Dense) | (None, 55, 32) | 2080 |
| max\_pooling2d\_3 (MaxPooling2D) | (None, 1, 4, 64) | 0 |
| dropout\_1 (Dropout) | (None, 55, 32) | 0 |
| flatten (Flatten) | (None, 256) | 0 |
| flatten\_1 (Flatten) | (None, 1760) | 0 |
| concatenate\_1 (Concatenate) | (None, 2016) | 0 |
| dense\_4 (Dense) | (None, 17) | 34289 |
| Total params: 3,696,785 | | |
| Trainable params: 3,694,865 | | |
| Non-trainable params: 1,920 | | |

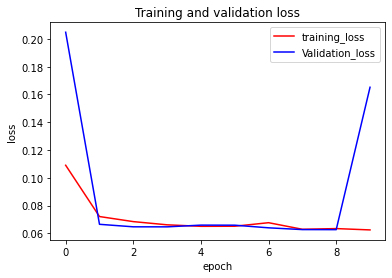
Table 5.10 The CNN-LSTM model architecture in detail

|  |  |
| --- | --- |
| **Parameters** | **Value** |
| Kernel size of Conv2D | (3, 3) |
| Kernel size of MaxPooling2D | (2, 2) |
| Fully connected layer | 128 |
| Activation function | ReLU |
| Padding method | same |
| Classification function | softmax |
| Optimizer | RMSprop |
| Epochs | 20 |
| Early stopping | 0.0001 |

Table 5.11 Parameters setting for CNN-LSTM model

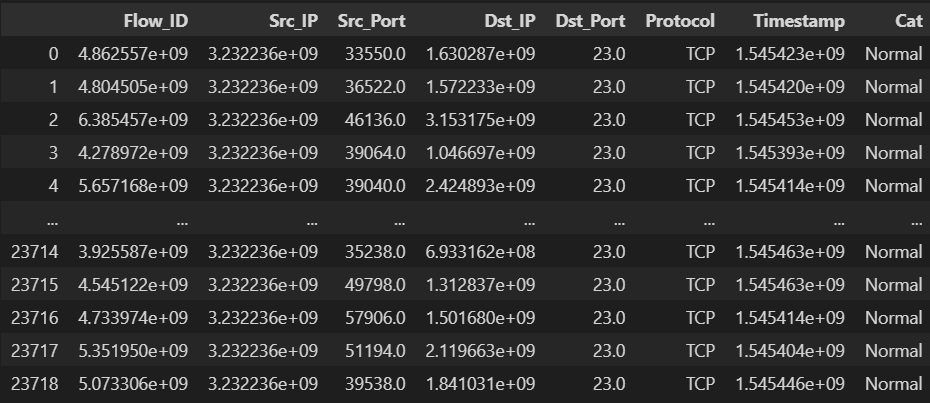
**5.3 Testing**

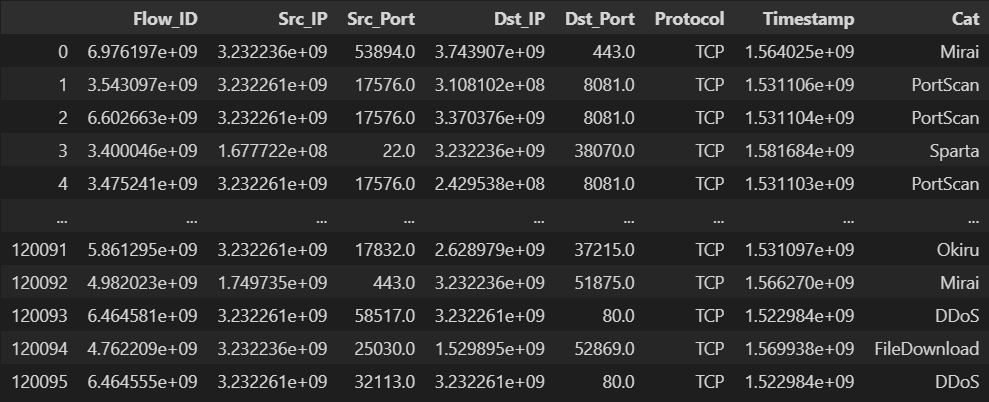




Accuracy Precision Recall F1\_Score TNR FPR NPV \ 0 0.989959 0.948281 0.990241 0.968807 0.989907 0.010093 0.998161 1 0.999861 0.999950 0.999554 0.999752 0.999981 0.000019 0.999826 2 0.999924 0.999477 0.999804 0.999641 0.999938 0.000062 0.999977 3 0.999924 0.999477 0.999804 0.999641 0.999938 0.000062 0.999977 4 0.999228 0.994306 0.997686 0.995993 0.999392 0.000608 0.999754 5 0.992602 0.917469 0.972011 0.943953 0.994012 0.005988 0.998075 6 0.999360 0.993451 0.995382 0.994415 0.999602 0.000398 0.999720 7 0.999465 0.994635 0.990111 0.992368 0.999805 0.000195 0.999640 8 0.999826 0.992283 0.997550 0.994909 0.999866 0.000134 0.999958 9 0.995828 0.883860 0.863219 0.873418 0.998077 0.001923 0.997682 10 0.999541 0.982699 0.984646 0.983671 0.999753 0.000247 0.999781 11 0.999771 0.986366 0.996292 0.991304 0.999817 0.000183 0.999951 12 0.999951 0.999272 0.995646 0.997455 0.999993 0.000007 0.999958 13 0.993158 0.917808 0.259891 0.405079 0.999790 0.000210 0.993350 14 0.992859 0.489559 0.207269 0.291235 0.998459 0.001541 0.994372 15 0.999708 0.990968 0.956413 0.973384 0.999951 0.000049 0.999755 16 0.999666 0.976898 0.945687 0.961039 0.999902 0.000098 0.999763

**5.4 Response**

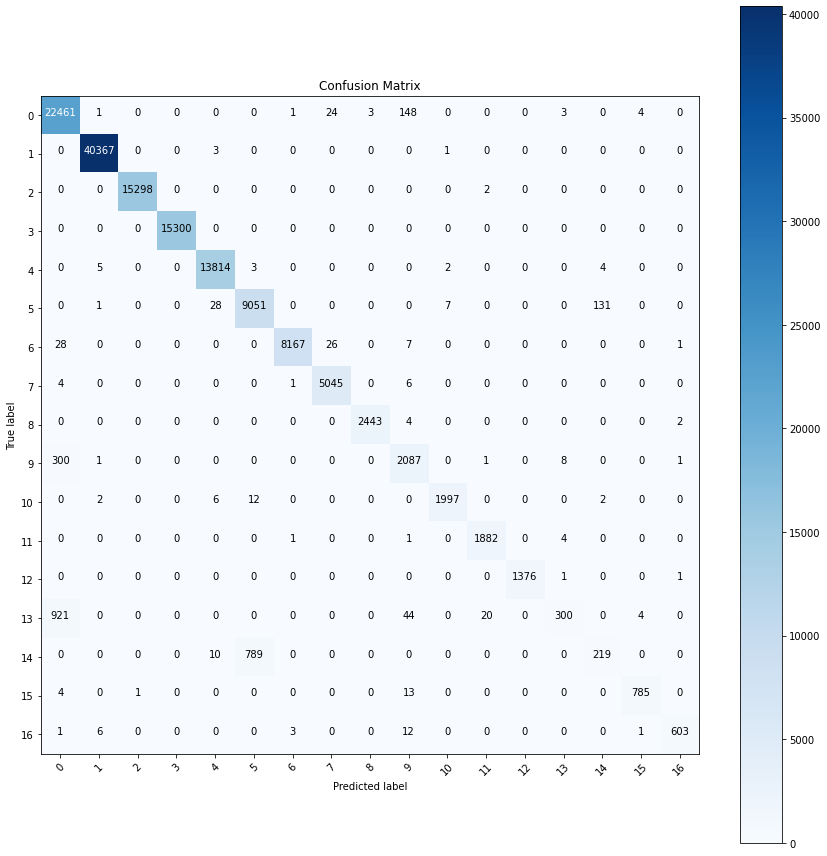


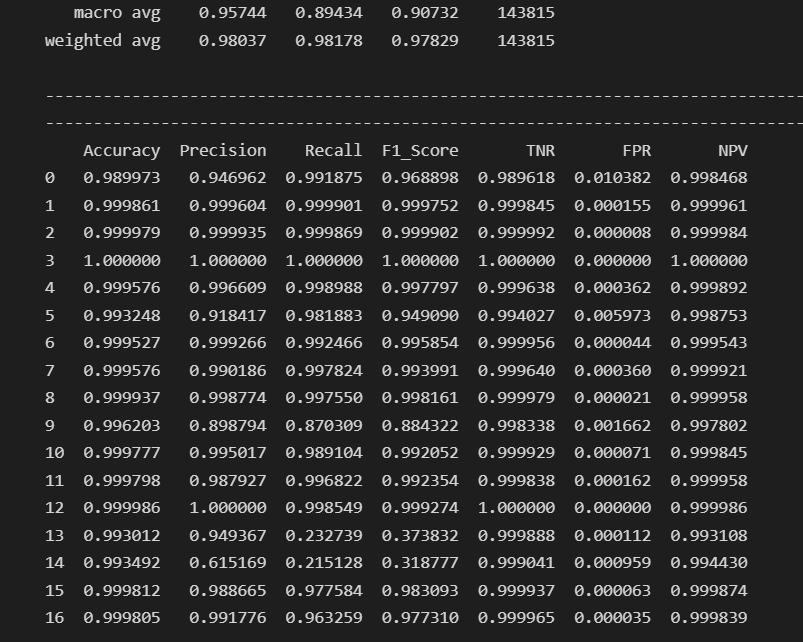


**Chapter 6**

**Experimental Results and Analysis**

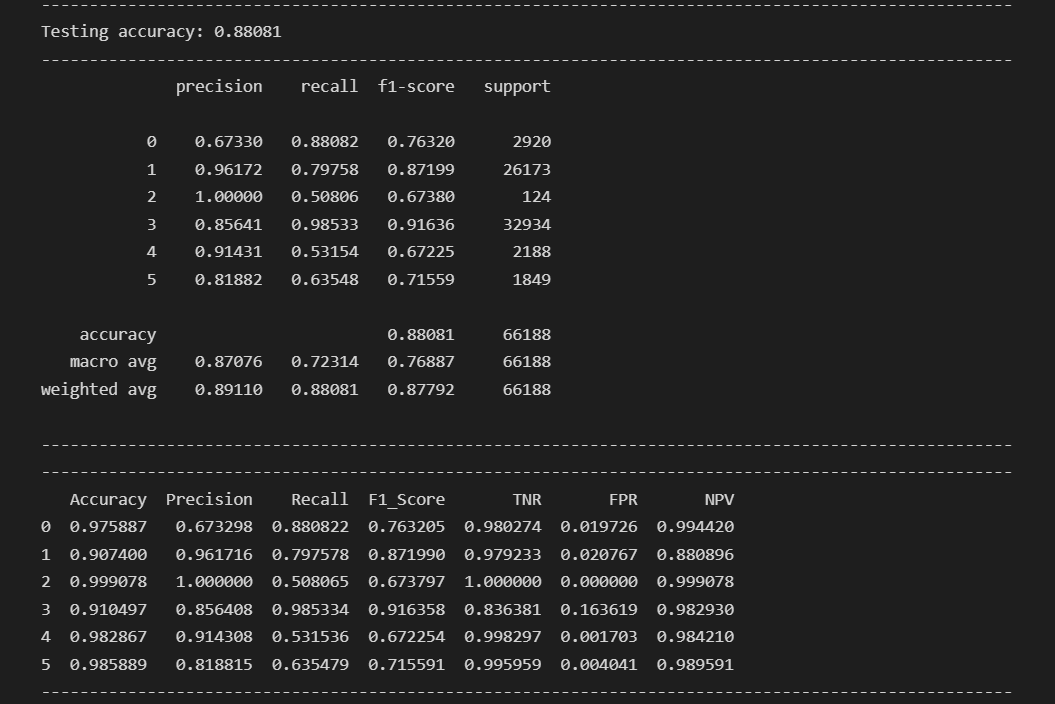
IoT-DS2



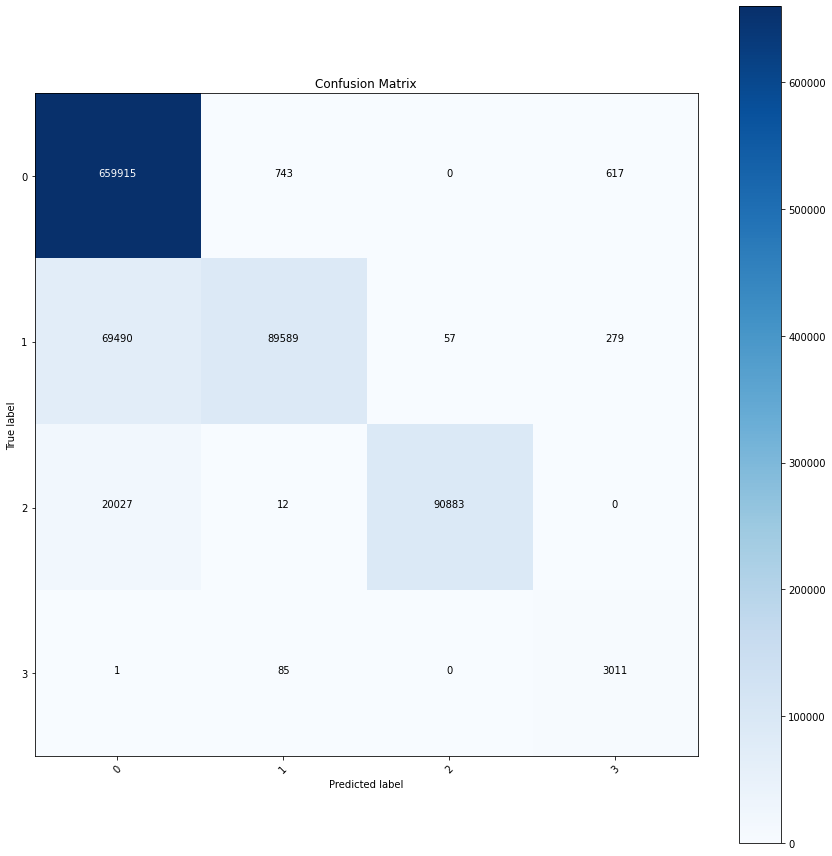


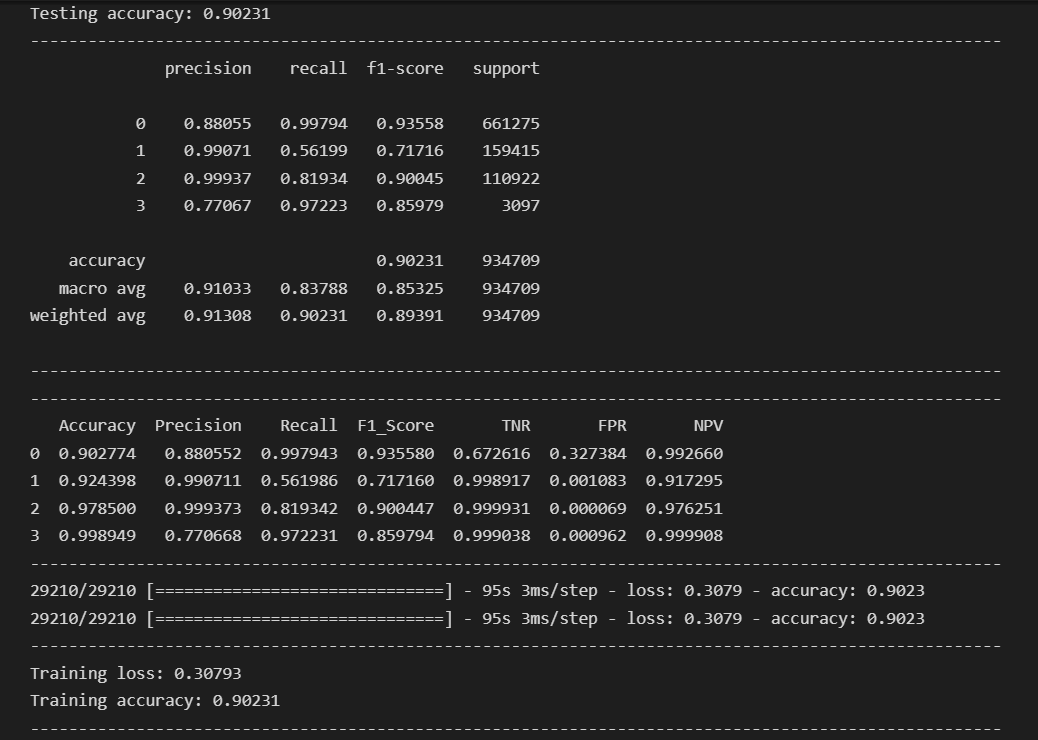
MQTTset Dataset





IoT-23 Dataset





|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Datasets** | **Accuracy** | **Precision** | **Recall** | **F1\_Score** | **TNR** | **FPR** | **NPV** |
| IoT-DS2 | 0.99791 | 0.98037 | 0.98178 | 0.97829 | 0.99880 | 0.00119 | 0.99890 |
| IoT-23 | 0.95115 | 0.91308 | 0.90231 | 0.89391 | 0.91762 | 0.08237 | 0.97152 |
| MQTTset | 0.88081 | 0.89110 | 0.88081 | 0.88081 | 0.96502 | 0.03497 | 0.97185 |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | Datasets | Training Time | Testing Time |  |
| CNN-LSTM | IoT-DS2 | 43m 10s | 5m 49s |  |
| CNN-LSTM | MQTTset |  |  |  |
| CNN-LSTM | IoT-23 | 11m 43s | 4m 36s |  |
|  |  |  |  |  |

**References**

1. Sinha, S. (2022). *State of IoT 2021: Number of Connected IoT Devices Growing 9% to 12.3 Billion globally, Cellular IoT Now Surpassing 2 Billion*. [online] IoT Analytics. Available at: <https://iot-analytics.com/number-connected-iot-devices/.>
2. Kim, S., Hwang, C. and Lee, T. (2020) ‘Anomaly based unknown intrusion detection in endpoint environments’, *Electronics (Basel)*, 9(6), pp. 1–21. doi:10.3390/electronics9061022.
3. Kirsch, C. (2016). *IDC: 70% of Successful Breaches Originate on the Endpoint | Rapid7 Blog*. [online] Rapid7. Available at: <https://www.rapid7.com/blog/post/2016/03/31/idc-says-70-of-successful-breaches-originate-on-the-endpoint/> [Accessed 8 Jun. 2022].
4. Chuvakin, A. (2013). *Named: Endpoint Threat Detection & Response.* [online] Availableat: <https://blogs.gartner.com/anton-chuvakin/2013/07/26/named-endpoint-threat-detection-response/> [Accessed 8 Jun. 2022].
5. Kaur, H. and Tiwari, R. (2021) ‘Endpoint detection and response using machine learning’, *Journal of Physics: Conference Series*, 2062(1), p. 12013. doi:10.1088/1742-6596/2062/1/012013.
6. Kumar, N., Kasbekar, G.S. and Manjunath, D. (2022) ‘Application of Data Collected by Endpoint Detection and Response Systems for Implementation of a Network Security System based on Zero Trust Principles and the EigenTrust Algorithm’.
7. Juma, N., Huang, X. and Tripunitara, M.V. (2020). Forensic Analysis in Access Control: Foundations and a Case-Study from Practice. *CCS*. doi:10.1145/3372297.3417860.
8. Cahill, D., & Poller, J. (2017). ‘An Adaptive and Layered Approach to Endpoint Security’, *The Enterprise Strategy Group*. [online] Available at: <https://www.bitdefender.co.th/media/wysiwyg/gravityzone/elite-security/ESG-White-Paper-Bitdefender-Jun-2017.pdf> [Accessed 9 Jun. 2022].
9. Badhwar, R. (2021). OEM and Third-Party Sourced Application and Services Risk.*TheCISO’s Next Frontier. Springer*, pp.335–344. doi:10.1007/978-3-030-75354-2\_41.
10. Yaffe, D. and Hendler, D. (2021) ‘Early Detection of In-Memory Malicious Activity Based on Run-Time Environmental Features’, *in Cyber Security Cryptography and Machine Learning. Cham: Springer International Publishing*, pp. 397–404. doi:10.1007/978-3-030-78086-9\_29.
11. Lee, S.-J., Shim, H.-Y., Lee, Y.-R., Park, T.-R., Park, S.-H. and Lee, I.-G. (2022). *Study on Systematic Ransomware Detection Techniques*. IEEE Xplore. doi:10.23919/ICACT53585.2022.9728909.
12. Scott, J. (2017). *Signature Based Malware Detection is Dead*. Institute for Critical Infrastructure Technology. Available at: <https://informationsecurity.report/Resources/Whitepapers/920fbb41-8dc9-4053-bd01-72f961db24d9_ICIT-Analysis-Signature-Based-Malware-Detection-is-Dead.pdf>.
13. Rakes, T.R., Deane, J.K. and Paul Rees, L. (2012) ‘IT security planning under uncertainty for high-impact events’, *Omega (Oxford)*, 40(1), pp. 79–88. doi:10.1016/j.omega.2011.03.008.
14. Ghafir, I. et al. (2018) ‘Detection of advanced persistent threat using machine-learningcorrelation analysis’*, Future generation computer systems*, 89, pp. 349–359. doi:10.1016/j.future.2018.06.055.
15. Aleroud, A. and Zhou, L. (2017) ‘Phishing environments, techniques, and countermeasures: A survey’, *Computers & security*, 68, pp. 160–196. doi:10.1016/j.cose.2017.04.006.
16. Jain, A.K. and Gupta, B.B. (2022) ‘A survey of phishing attack techniques, defence mechanisms and open research challenges’, *Enterprise information systems*, 16(4), pp.527–565. doi:10.1080/17517575.2021.1896786.
17. Kumar, M.S., Ben-Othman, J., Srinivasagan, K.G. and Krishnan, G.U. (2019). *Artificial Intelligence Managed Network Defense System against Port Scanning Outbreaks*. IEE Xplore. doi:10.1109/ViTECoN.2019.8899380.
18. Sikder, A.K. et al. (2021) ‘A Survey on Sensor-Based Threats and Attacks to Smart Devices and Applications’, *IEEE Communications surveys and tutorials*, 23(2), pp. 1125–1159. doi:10.1109/COMST.2021.3064507.
19. Šemić, H. and Mrdovic, S. (2017). *IoT honeypot: A multi-component solution for handling manual and Mirai-based attacks*. IEEE Xplore. doi:10.1109/℡FOR.2017.8249458.
20. Gandotra, E., Bansal, D. and Sofat, S. (2014) ‘Malware Analysis and Classification: A Survey’, *Journal of information security*, 5(2), pp. 56–64. doi:10.4236/jis.2014.52006.
21. Yang Xin et al. (2018) ‘Machine Learning and Deep Learning Methods for Cybersecurity’, *IEEE access*, 6, pp. 35365–35381. doi:10.1109/ACCESS.2018.2836950.
22. Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT ’92*. Association for Computing Machinery, 144-152. doi:10.1145/130385.130401.
23. Gonzalez-Abril, L. et al. (2017) ‘Handling binary classification problems with a priority class by using Support Vector Machines’, *Applied soft computing*, 61, pp. 661–669 doi:10.1016/j.asoc.2017.08.023.
24. Quinlan, J.R. (1986) ‘Induction of decision trees’, *Machine learning*, 1(1), pp. 81–106 doi:10.1023/A:1022643204877.
25. Quinlan, J.R. et al. (1993) *C4.5 : programs for machine learning*. San Mateo, California: Morgan Kaufmann Publishers.
26. Breiman, L. (1984) *Classification and regression trees.* Boca Raton, Florida: Chapman & Hall : CRC.
27. Tin Kam Ho (1995). *Random decision forests*. [online] IEEE Xplore. doi:10.1109/ICDAR.1995.598994.
28. Ambwani, T. (2003) ‘Multi class support vector machine implementation to intrusion detection’, in *Proceedings of the International Joint Conference on Neural Networks, 2003.* IEEE, pp. 2300–2305 vol.3. doi:10.1109/IJCNN.2003.1223770.
29. Ronghua Tian et al. (2010) ‘Differentiating malware from cleanware using behavioural analysis’, in *2010 5th International Conference on Malicious and Unwanted Software.* IEEE, pp. 23–30. doi:10.1109/MALWARE.2010.5665796.
30. Lee, Y.-J. and Mangasarian, O.L. (2001). RSVM: Reduced Support Vector Machines. *Proceedings of the 2001 SIAM International Conference on Data Mining*. pp.1-17doi:10.1137/1.9781611972719.13.
31. Chiu, C.-Y. et al. (2010) ‘Semi-supervised Learning for False Alarm Reduction’, in *Advances in Data Mining. Applications and Theoretical Aspects.* Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 595–605. doi:10.1007/978-3-642-14400-4\_46.

1. Ioannou, C. and Vassiliou, V. (2019). Classifying Security Attacks in IoT Networks Using Supervised Learning. *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 652-658. doi:10.1109/dcoss.2019.00118.
2. Ioannou, C. and Vassiliou, V. (2021) ‘Network attack classification in iot using support vector machines’, *Journal of sensor and actuator networks*, 10(3), p. 58. doi:10.3390/jsan10030058.
3. Hasan, M. et al. (2019) ‘Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches’, *INTERNET OF THINGS*, 7, p. 100059. doi:10.1016/j.iot.2019.100059.
4. M. Pahl and F. Aubet. (2018) ‘All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection’, *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 72-80.
5. Alothman, Z., Alkasassbeh, M. and Al-Haj Baddar, S. (2020) ‘An efficient approach to detect IoT botnet attacks using machine learning’, *Journal of high speed networks*, 26(3), pp. 241–254. doi:10.3233/JHS-200641.
6. Varma P, R.K., Mallidi, S.K.R. and Muni, R.R. (2021). A Wrapper based Feature Selection using Grey Wolf Optimization for Botnet Attack Detection. *International Journal of Sensors, Wireless Communications and Control*, 11(9). pp .951-956. doi:10.2174/2210327911666210120124340.
7. Ongun, T. *et al.* (2019) ‘On Designing Machine Learning Models for Malicious Networ k Traffic Classification’.
8. Machaka, P., Ajayi, O., Maluleke, H., Kahenga, F., Bagula, A. and Kyamakya, K. (2021). Modelling DDoS Attacks in IoT Networks using Machine Learning. *arXiv:2112.05477 [cs]*.
9. Abiodun, O.I. *et al.* (2018) ‘State-of-the-art in artificial neural network applications: A survey’, *Heliyon*, 4(11), pp. e00938–e00938. doi:10.1016/j.heliyon.2018.e00938.
10. Huang, T.-J. (2017) ‘Imitating the Brain with Neurocomputer A ＂New＂ Way Towards Artificial General Intelligence’, *International journal of automation and computing*, 14(5), pp. 520–531. doi:10.1007/s11633-017-1082-y.
11. Xin, Y. et al. (2018) ‘Machine Learning and Deep Learning Methods for Cybersecuri y’, *IEEE access*, 6, pp. 35365–35381. doi:10.1109/ACCESS.2018.2836950.
12. Gang Liu, Fen Hu and Wei Chen (2010) ‘A neural network ensemble based method for detecting computer virus’, in *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering.* IEEE, pp. 391–393. doi:10.1109/CMCE.2010.5610520.
13. Nagisetty., A., & Gupta, G.P. (2019). Framework for Detection of Malicious Activities in IoT Networks using Keras Deep Learning Library. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 633-637.
14. Ji, S.-Y. et al. (2016) ‘A multi-level intrusion detection method for abnormal network behaviors’, *Journal of network and computer applications*, 62, pp. 9–17. doi:10.1016/j.jnca.2015.12.004.
15. Donghong, S. et al. (2014) ‘Analysis of Network Security Data Using Wavelet Transforms’, *Journal of algorithms & computational technology*, 8(1), pp. 59–70. doi:10.1260/1748-3018.8.1.59.
16. Li, H. and Ding, X. (2017) ‘Research on artificial neural network intrusion detection photochemistry based on the improved wavelet analysis and transformation’, in *AIP Conference Proceedings. Melville: American Institute of Physics*. doi:10.1063/1.4977374.

1. Gao, M. et al. (2020) ‘Malicious network traffic detection based on deep neural networks and association analysis’, *Sensors (Basel, Switzerland)*, 20(5), p. 1452. doi:10.3390/s20051452.
2. Cheng, Y. et al. (2018) ‘Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges’, *IEEE signal processing magazine*, 35(1), pp. 126–136. doi:10.1109/MSP.2017.2765695.
3. Lee, W. Y., Saxe, J., & Harang, R. (2019). SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks. In *Deep learning applications for cyber security*  pp. 197-210.
4. Tavallaee, M. *et al.* (2009) ‘A detailed analysis of the KDD CUP 99 data set’, in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, pp. 1–6. doi:10.1109/CISDA.2009.5356528.
5. Khan, Muhammad Almas et al. (2021) ‘A deep learning-based intrusion detection system for mqtt enabled iot’, *Sensors (Basel, Switzerland)*, 21(21), p. 7016. doi:10.3390/s21217016.
6. Hindy, H. et al. (2020) ‘Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset)’.
7. Ullah, I. and Mahmoud, Q.H. (2022) ‘An Anomaly Detection Model for IoT Networks based on Flow and Flag Features using a Feed-Forward Neural Network’, in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, pp. 363–368. doi:10.1109/CCNC49033.2022.9700597.
8. Vaccari, I. et al. (2020) ‘MQTTset, a New Dataset for Machine Learning Techniques on MQTT’, *Sensors (Basel, Switzerland)*, 20(22), p. 6578. doi:10.3390/s20226578.
9. Shalaginov, Andrii, Semeniuta, Oleksandr and Alazab, Mamoun (2019), MEmachine learning: Resource-Aware MQTT-Based Machine Learning for Network Attacks Detection on IoT Edge Devices. *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion - UCC ’19 Companion.* pp. 123-128.
10. Hariprasad Siddharthan, T. Deepa and Prabhu Chandhar (2022) ‘SENMQTT-SET: An Intelligent Intrusion Detection in IoT-MQTT Networks Using Ensemble Multi Cascade Features’, *IEEE access*, 10, pp. 33095–33110. doi:10.1109/ACCESS.2022.3161566.
11. Idrissi, Idriss, Azizi, Mostafa and Moussaoui, Omar (2022), An Unsupervised Generative Adversarial Network Based-Host Intrusion Detection System for Internet of Things Devices. *Indonesian Journal of Electrical Engineering and Computer Science*, 25(2): 1140.
12. Koroniotis, N. et al. (2019) ‘Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset’, *Future generation computer systems*, 100, pp. 779–796. doi:10.1016/j.future.2019.05.041.
13. Shafiq, M. et al. (2020) ‘Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city’, *Future generation computer systems*, 107, pp. 433–442. doi:10.1016/j.future.2020.02.017.
14. Abdalgawad, N. et al. (2022) ‘Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset’, *IEEE access*, 10, pp. 6430–6441. doi:10.1109/ACCESS.2021.3140015.
15. Oha, C.V. et al. (2022) ‘Machine Learning Models for Malicious Traffic Detection in IoT Networks /IoT-23 Dataset’, in *Machine Learning for Networking.* Cham: Springer International Publishing, pp. 69–84. doi:10.1007/978-3-030-98978-1\_5.
16. Liu, Zhipeng, Thapa, Niraj, Shaver, Addison, Roy, Kaushik, Yuan, Xiaohong and Khorsandroo, Sajad (2020), Anomaly Detection on IoT Network Intrusion Using Machine Learning. *IEEE Xplore*: pp. 1–5.
17. Labs, Arkose (2020), Arkose Labs Reveals 20% Spike in Fraud as Digital Behavior Shifts during COVID-19. *GlobeNewswire News Room*. available at https://www.globenewswire.com/news-release/2020/04/28/2023282/0/en/Arkose-Labs-Reveals-20-Spike-in-Fraud-as-Digital-Behavior-Shifts-During-COVID-19.htmachine learning [14 July 2022].
18. Abdullah Alsaedi et al. (2020) ‘TON\_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems’, *IEEE access*, 8, pp. 165130–165150. doi:10.1109/ACCESS.2020.3022862.
19. Khan, Muhammad Almas et al. (2021) A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT, *Sensors (Basel, Switzerland)*, 21(21), p. 7016. doi:10.3390/s21217016.
20. Ullah and Q. H. Mahmoud (2021), Ontario Tech University. *sites.google.com*. available at <https://sites.google.com/view/iotdataset1>