Proceedings of the 8th International Conference on Computing Methodologies and Communication (ICCMC-2025)
DVD Part Number: CFP25K25-DVD; ISBN: 979-8-3315-1210-1

LSTM-Based Malware Classification Using Bytecode and Opcode Features for IoT Security

M. Subhameena ,
Department of CSE,
VISTAS,
Chennai-600117,TamilNadu,India
subhameena.m@gmail.com

A.Rajesh ,
Department of CSE,
VISTAS,
Chennai-600117, Tamil Nadu, India,
arajesh.se@vistas.ac.in

A.Packialatha
Department of CSE,
VISTAS,
Chennai-600117,TamilNadu,India,
Packialatha.se@vistas.ac.in

Thilakavathy P,
Department of CSE,
VISTAS,
Chennai-600117,TamilNadu,India
thilakavathy.se@vistas.ac.in

A.Banushri
Department of CSE,
VISTAS,
Chennai-600117, TamilNadu, India
Banushri.se@vistas.ac.in

Abstract

This study proposes a deep learning approach for malware classification, utilizing both byte-level and opcode features. The dataset comprises the byte files and opcode sequences, which are preprocessing, features are extracted, and integration with corresponding class labels. Byte files are analysed for size distribution, address removal, and unigram bag-of-words representation, while opcode features are extracted separately. SMOTE is used to tackle the challenge of class imbalance An LSTM neural network is employed to categorized the malware into distinct families by using the extracted features. This model is designed to differentiate between various malware families, including Ramnit (Trojan), Lollipop (Adware), Vundo (Trojan), Simda (Backdoor). Finally, This model is trained with categorical cross entropy loss and optimized using the Adam optimizer is utilized to enhanced and improved the accuracy across the different malware types. The proposed methodology achieves a test accuracy of 86.5%, delivers with strong classification results as precision (87.8%), recall (91.2%), and F1-score (85.0%). The confusion matrix analysis indicates minimal misclassifications among malware families, demonstrating the model's robustness. Performance metrics used to validate and confirm the model's effectiveness in generalization, demonstrated y analyzing training and validation accuracy and loss curves showing the stable convergence.

Keywords*: IOT, File size based, Byte value, Dataset, preprocessing data, Malware detection, Benign, Opcode Category, Feature Extraction, LSTM classification, LSTM model, SMOTE, Test, Train, Evaluation, and Accuracy.*

1. INTRODUCTION

Role of IoT: IoT (Internet of Things) malware detection refers to identifying and classifying malicious software specifically targeting IoT devices, including smart home gadgets, industrial control systems, and wearable technology. Given the growing number of IoT devices, Cybercriminals often target these systems due to their vulnerabilities is due to their limited security capability and frequent lack of regular updates. The methodology of extracting meaningful features from bytecode and opcode sequences can be applied to IoT malware binaries [14]. This includes analysing unique patterns in IoT firmware files and identifying common attack signatures. Unlike traditional malware, IoT malware often focuses on remote control, botnet formation (e.g., Mirai), and (DDoS) attacks

The model can be trained on IoT-specific malware datasets to enhance detection. The trained model can be optimized for real-time detection and deployed on IoT gateways or edge devices, providing an additional security layer without relying on cloud-based detection.

Malware detection and classification are crucial to mitigating these risks. Traditional signature-based detection methods often struggle to identify emerging and evolving threats, as they rely on known patterns and previously encountered malware signatures. This limitation makes the application of machine learning and deep learning techniques essential for effectively detecting and classifying new and sophisticated malware variants. These advanced methods can learn complex patterns from data, enabling them to generalize beyond known signatures and identify previously unseen threats. In the context of IoT malware, deep learning models—especially those leveraging opcode and bytecode features—offer a dynamic and scalable approach to enhance security across diverse IoT environments.

The proposed system is designed for IoT detecting and classifying malware according to opcode categories. Explanation of opcodes (operation codes) as low-level instructions executed by the CPU [15].

The role of opcode categories in reflecting malware behavior, including arithmetic, logic, control, and data transfer operations. Previous studies on opcode usage in malware classification. The classification of opcode sequences into categories that highlight malware behavior patterns. Impact of opcode categories on the representation of malware behavior. This study focuses on building a robust deep learning model for malware classification using LSTM networks. The dataset consists of byte files and assembly opcode features extracted from malware binaries.

The project leverages feature engineering techniques, data preprocessing, and advanced machine learning methodologies to improve accuracy of malware detection. The dataset is acquired from malware samples, containing byte files and opcode features. Preprocessing steps include removing unwanted data, normalizing numerical values, and handling missing

values. The IoT malware in binary form is transformed into a series of opcodes, with every opcode assigned to a category based on its functionality. Features are extracted utilizing common n-grams, 18,856 maximum sub- patterns, and corresponding entropy values. The IoT malware dataset is analysed using various techniques, including file details and organization during packing. We provided a detailed explanation of the feature creation by using the opcode categories to effectively represent the characteristics of IoT malware.

The features were visualized, so common patterns and differences were observed Extract meaningful features from byte files using custom-built bag-of-words techniques and opcode unigram representations to better represent malware characteristics. Since malware datasets often exhibit class imbalance, the Synthetic Minority Over-sampling Technique SMOTE is used to balance the dataset and improve model generalization. LSTM network, a type of Recurrent Neural Network (RNN), for malware classification. LSTM Layers is used to process sequential data (byte/opcode sequences) and capture long-term dependencies in malware patterns. Design and train an LSTM deep learning models for malware classification. This model processes sequences of extracted features to classify malware into different categories. Evaluate The trained model evaluated using metrics like accuracy,confusion matrices, and loss functions to ensure effective malware detection. Save the trained model for potential deployment and explore improvements, such as hyperparameter tuning and integrating additional deep learning architectures. By implementing deep learning techniques for malware classification, this project seeks to enhance the accuracy and effectiveness of detection methods new and sophisticated malware, thereby enhancing cybersecurity Défense mechanisms. Confusion Matrix evaluates classification performance by visualizing true vs. predicted labels [16].

Summary: This study addresses these challenges by applying deep learning techniques to detect and classify IoT malware. By analyzing byte-level and opcode-level features extracted from malware binaries, the proposed system can identify patterns specific to IoT-targeting threats. The model is designed to be lightweight enough for deployment on IoT edge devices or gateways, allowing for real-time local threat detection without depending on cloud infrastructure. This makes it a practical and scalable solution for enhancing IoT security.

2. RELATED WORK

Lee, Hyun Jong, Soin Kim, Dong Heon Baek, Dong Hoon Kim, and Doo sung Hwang et.al.,[1] proposed that “Robust IoT Malware Detection and Classification Using Opcode Category Features on Machine Learning” states that IoT malware samples and benign software samples are collected from sources like IoT malware repositories and real-world IoT applications. The main idea is to analyse the opcode sequences in binary executables, instead of using raw opcodes, opcode categories are used to extract meaningful patterns. Opcode category frequency distributions are used as feature vectors. Statistical methods and normalization techniques are implemented to optimize the extracted opcode features. The algorithms are tested utilizing evaluation measures such as accuracy, precision, recall,F1 score, and ROC AUC help in analysing the model’s performance.

The method relies on static analysis, meaning obfuscation techniques like polymorphic and metamorphic malware can evade detection. Extracting opcode category features from large IoT malware datasets can be computationally expensive. Since the approach is based on known opcode patterns, it may not be very effective against novel, unseen malware variants. Performance may degrade when applied to new datasets or different IoT environments due to variations in compiled binaries.

X. Jiang et. al.,[2] have proposed that a An experimental analysis of industrial security vulnerabilities IoT devices." States that this paper mainly aimed at identifying and analyzing the security vulnerabilities in Industrial IoT devices through experimental testing. The various IoT devices are chosen based on their popularity, usage in critical infrastructure, and known security concerns. The devices include industrial sensors, controllers (PLC/SCADA), smart gateways, and edge computing devices. The attack surface of each IoT device is mapped, identifying all possible entry points for attacks (e.g., network interfaces, APIs, firmware, web applications, and hardware ports).Static analysis ,dynamic analysis and Fuzz testing is used for vulnerability assessment process. Exploits are executed in a controlled lab environment to assess their impact. Security patches, configuration changes, and best practices are proposed to mitigate identified vulnerabilities. The effectiveness of security measures is evaluated post-mitigation finally repeat the penetration tests are conducted to verify if vulnerabilities persist. Limited Device Coverage ,Controlled Lab Environment Ethical and Legal Constraints ,rapidly Evolving Threat Landscape: Device-Specific Exploits are the limitations.

K. Albulayhi, et.al.,[3] have proposed that IoT intrusion detection using machine learning with a novel high performing feature selection method “states that this paper focuses on enhancing IoT intrusion detection systems (IDS) using machine learning (ML) and an optimized feature selection method to improve detection accuracy and efficiency. dataset collected, data cleaning process then Feature encoding by categorical features are converted into numerical values. Normalization like Feature values are scaled to improve ML model performance. A new feature selection method is proposed to identify the most relevant features for intrusion detection. The process involves Filter Methods, Wrapper Methods, Hybrid Approach. The selected features aim to reduce dimensionality while maintaining high classification accuracy. Dataset Bias, Performance may degrade when applied to new IoT environments with different network traffic patterns. Feature selection and ML model training may require high computational power, especially for deep learning approaches and scalability are the limitations.

J. Paik, et.al., [4] proposed that “Malware classification using a byte granularity feature based on structural entropy” states that This paper presents a malware classification approach using structural entropy at the byte level to capture malware characteristics more effectively A dataset of malware samples is gathered from sources and the samples are categorized based on malware families (e.g., trojans, worms, ransomware). Benign executable files are also collected for comparison Entropy measures randomness in a system; malware often has distinguishable entropy patterns in different sections of the binary. The binary file is divided into fixed-size byte segments Malware often contains compressed or encrypted sections, leading to high entropy in specific regions. This entropy distribution is used as a feature for classification. The extracted entropy sequences are transformed into feature vectors. he trained models are tested on a separate dataset using metrics like Accuracy, Precision, Recall, F1-score,Confusion Matrix, Obfuscation & Packing Techniques, Loss of Contextual Information, Computational Overhead, Generalization to Unknown Malware Dataset Bias .The classification model's accuracy depends on the diversity of the dataset used for training

S. Moon, Y. Kim, H. Lee, D. Kim, D. Hwang et.al.,[5] proposed that “Evolved IoT malware detection using opcode category sequence through machine learning’’ states that This paper proposes an enhanced IoT malware detection approach using opcode category sequences instead of raw opcodes, making detection more robust and efficient. A dataset of IoT malware binaries is gathered from public repositories the dataset is preprocessed to remove duplicate and corrupted files. The malware binaries are disassembled using tools like IDA Pro, Radare2.Instead of using raw opcodes (which vary across architectures), opcodes are grouped into functional categories such as Arithmetic operations (add, sub, mul)and logical operations. This transformation generalizes the feature set, making the model more adaptable across different CPU architectures. The ordered sequence of opcode categories is extracted from each binary n-gram modelling is used to capture local patterns within the sequence. TF-IDF (Term Frequency-Inverse Document Frequency) is applied to opcode sequences to prioritize important pattern. The models are evaluated using accuracy, precision, recall, and F1- score. Obfuscation and Evasion Techniques, Architecture Dependence, Computational Overhead ,Limited Generalization to Unknown Malware ,Lack of Behavioural Context are the limitations[12].

Ahsan Nazira, Jingsha Hea, Nafei Zhua, Saima Siraj Qureshia, Siraj Uddin Qureshia, Faheem Ullaha, Ahsan Wajahata, Muhammad Salman Pathanb,c,et,al.,[6]”proposed that deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem” states that This paper presents an Integrated a deep learning methodology that leverages a combination of CNN and LSTM for enhanced performance for identifying threats in IoT in IoT environments efficiently. The dataset includes both benign and malicious network traffic, covering various attack types as Denial of Service (DoS) attacks ,Man-in-the-Middle (MitM) attacks, Botnets and malware infections. Data Preprocessing & Feature Engineering Extracting relevant features such as packet size, protocol type, source/destination IP, and flow duration. CNN for Spatial Feature Extraction. The CNN extracts spatial patterns from network traffic data .It captures correlations between different traffic features using convolutional layers. The LSTM component models sequential dependencies in network traffic data. It detects patterns over time, making it effective for identifying slow and stealthy attacks. Model Training & Optimization by using loss Function Categorical Cross-Entropy for multi-class classification. This model is evaluated using key metrics as Accuracy, Precision, Recall, and F1-score and Confusion Matrix. The limitations of this approach is High Computational Cost, Data Dependence & Generalization Potential Overfitting Difficulty in Real- Time Deployment, Limited Interpretability The CNN- LSTM hybrid model effectively detects IoT threats by effectively capturing both spatial and temporal features in network traffic

Maran, Piragash, Timothy Tzen Vun Yap, Ji Jian Chin, Hu Ng, Vik Tor Goh, and Thiam Yong Kuek.et.,al[7] proposed that "Comparison of machine learning models for IoT malware classification “states that This paper evaluates different machine learning (ML) models for classifying IoT malware, analysing their effectiveness based on feature selection, model performance, and detection accuracy. IoT malware samples are collected from sources like VirusTotal, IoTPOT, and MalwareBazaar. Benign IoT applications are also included for balanced classification. Features are extracted from malware binaries and network traffic. The two main types of features are considered as Static Features Dynamic Features [13] .Dimensionality reduction is applied to select the most relevant features. Principal Component Analysis (PCA), Mutual Information, and Recursive Feature Elimination (RFE) are utilized to optimize performance. Models are trained using training datasets and validated using a hold-out test set or cross- validation (e.g., k-fold validation).Evaluation metrics include Accuracy, Precision, Recall, and F1-score,False Positive Rate (FPR)Limitations of the Approach are Dataset Bias ,Computational Cost, Feature Dependence, Generalization Issues, Class Imbalance. This study provides insights into the strengths and weaknesses of different ML models for IoT malware classification. XGBoost and deep learning models offer high accuracy, but traditional models like Random Forest provide a better trade-off between efficiency and accuracy

Riaz, Sharjeel, Shahzad Latif, Syed Muhammad Usman, Syed Sajid Ullah, Abeer D. Algarni, Amanullah Yasin, Aamir Anwar, Hela Elmannai, and Saddam Hussain at.,al[8] "Malware detection in internet of things (IoT) devices using deep learning” states that This paper presents a deep learning-based approach for detecting malware in IoT devices, leveraging neural networks to analyze malware behaviors and patterns. IoT malware and benign samples are collected from repositories like VirusTotal, IoTPOT, MalwareBazaar, and IoT-23 dataset. The dataset is pre-processed to balance benign and malicious samples to avoid classification bias. Feature extraction is performed using static analysis, dynamic analysis, or network traffic analysis. Data Preprocessing Normalization & Encoding: Features are normalized to a fixed scale (e.g., Min-Max Scaling).One-Hot Encoding Categorical features (e.g., API calls, opcodes) are converted into vector representations and Feature Reduction. CNNs are used to extract spatial patterns from feature matrices (e.g., opcode sequences or system call logs).LSTM networks process time-series data, making them effective for analysing sequential patterns in malware behaviour. Model Training & Optimization, Performance Evaluation, then the Metrics Used to find out the Accuracy, Precision, Recall, and F1-score, False

Positive Rate (FPR).The Limitations of the Approach are Computational Cos, Dataset Bias & Generalization Issues

, Obfuscation & Evasion Techniques, Real-Time Detection Challenges, Interpretability & Explainability

Mehrban, Ali, and Pegah Ahadian.et al[9] proposed that" Malware detection in iot systems using machine learning techniques" states that a hybrid a deep learning model that combines CNN and LSTM for enhanced feature extraction and sequence learning to enhance malicious software detection in Internet of Things IoT environments. The researchers compiled a dataset comprising both benign and malicious IoT software samples. Before training the model, the dataset was preprocessed through various steps such as normalization and encoding to ensure compatibility with the deep learning models. These steps aimed to standardize the data and convert categorical variables into numerical formats suitable for analysis. Employed to derive spatial features from the input data. Capturing local patterns within the malware samples utilized to recognize temporal relationships and sequential trends in the data, which are crucial to comprehend the behavior of malicious software over time. The hybrid model was trained using a K-fold cross-validation technique to ensure robustness and mitigate overfitting. Performance was evaluated based on accuracy, with the proposed model achieving a 95.5% accuracy rate, outperforming existing methods. Computational Complexity, Dataset Limitations, Feature Extraction Dependency and Real- Time Detection Challenges are the limitations.

3. PROPOSED METHODOLOGY

The proposed approach leverages Deep Learning techniques, particularly a (LSTM) neural network, to efficiently malware detection and malware classification. The system processes raw .bytes files and .asm opcode sequences used to extract relevant features that represent malware behaviour. A custom feature engineering approach is implemented by including unigram frequency analysis of hexadecimal bytes and opcode sequences. To address class imbalance issues, SMOTE (Synthetic Minority Over-sampling Technique) is used to enhance underrepresented malware families, ensuring the model learns from a well-balanced dataset. The extracted features are then normalized and converted into structured numerical input, optimizing them suitable for deep learning-based classification.

The LSTM-based model is designed to identify the sequential patterns in malware binaries, enabling it to distinguish between different malware families with high precision The architecture comprises an LSTM layer featuring with 128 components which is succeeded by a dropout A tier (0.5) to prevent overfitting, and dense layers with ReLU activation to capture hierarchical patterns effectively. The final classification is achieved and performed using a SoftMax activation function, predicting one of the nine malware families. The training process of the model utilizes the Adam optimizer and categorical cross entropy loss for attaining a good accuracy with 81.5%.This approach underscores the efficiency of utilizing the deep learning for identifying the malware offered the scalable and robust solution for cybersecurity applications.

The proposed model key procedures:
(1) Feature Extraction
(2) Preprocessing of Data
(3) Build and LSTM model
(4) LSTM based Deep Learning Model for classification
(5) Evaluation Metrics

Architecture diagram

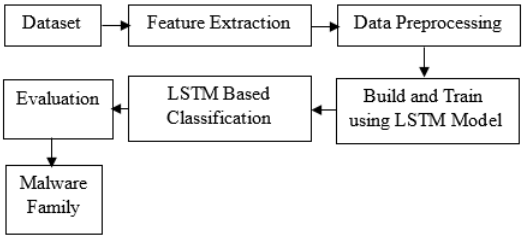


Figure 1.1 System Architecture

The system architecture diagram will explain stuffs like how the modules are being connected, how the data is transferring within them, how the modules are placed. Figure 1.1 explains the hierarchy order of each module used in this project. The system architecture will also be used while debugging the project. If any kind of error occurred during the compilation of the project or while during the run time of the project, it would be easy to point out the error raising module if we use the system architecture of a project. Figure 1.3 explains about the Data Flow diagram of our research.This makes the debugging part to be easier and more adequate. This contains all the glimpse or an overview of the modules which are used in this project.

3.1 Dataset

A Malware Dataset is a compilation of data samples associated with malicious software (malware). It usually contains the traits, behaviours, and attributes of various malware types, which can be utilized for research, machine learning, or cybersecurity analysis. The malware dataset is sourced from the Microsoft Malware Classification Challenge (BIG 2015), hosted on Kaggle. The dataset contains ~1,00,000 malware samples.

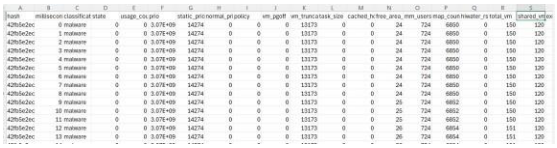


Figure 1.2 Malware dataset

Each malware has two files .bytes file which has raw hexadecimal representation and .asm file has assembly code (disassembled from the executable). The malware dataset Figure 1.2 consists of malware samples in the form of disassembled bytecode (.bytes files) and assembly (.asm files), along with labels specifying the malware family. The dataset includes trainLabels.csv contains the ID of each malware sample and its corresponding Class (1-9) in Table 1.1.In Byte Files (.bytes) files contain the hexadecimal representation of executable files, without headers. The first column is the memory address, which is typically removed during preprocessing. In Assembly Files (.asm), These contain the disassembled assembly code of the malware samples.

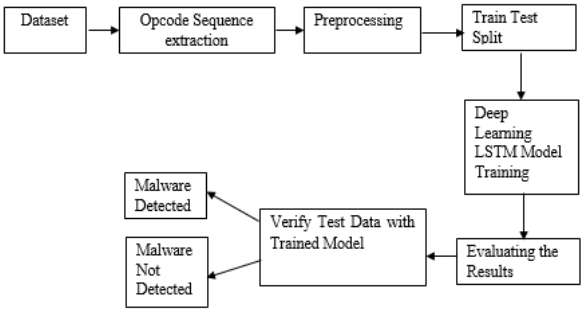


Figure 1.3 Data Flow diagram

ID	Class
0A32eTdBKayjCWhZqDOQ	2
3c0e22f2a22a67c8b083	3
8dc28f4826df79a1e80f	1

Table 1.1 Malware ID and class

3.2 Feature extraction

Feature extraction entails involves converting raw data into a structured set of meaningful and relevant features, facilitating machine learning and data analysis tasks and useful features, enabling machine learning and data analysis tasks. It reduces data dimensionality while preserving essential information.

Figure 1.4 Opcode 40 Feature extraction

3.2.1 File Size Based

In File Size-Based Feature Extraction, The size of each .bytes file is extracted and stored as a feature. The os.stat() function gets the file size. The size is converted into MB and stored in a Data Frame. Malware families often have distinct file sizes based on their functionality.

3.2.2 Byte Value Frequency

In Byte Value Frequency (Bag of Words Representation), Counts of each hexadecimal byte (00 to FF) in the .bytes file. Count of unknown bytes (??), which may represent missing or obfuscated values. First Initialize a 257- column matrix which is 256 columns for byte values (00- FF).and 1 column for ?? (unknown bytes),then Iterate through .bytes files: Open and read each line, next by removing the memory addresses, Count the occurrences of each hex value (00-FF) . Finally Store in feature_matrix. At last Save the features to result.csv. Byte frequency analysis helps detect patterns that are unique to certain malware families

3.2.3 Opcode Feature Extraction

An opcode is a component of a machine instruction that defines the specific operation to be executed. (e.g., MOV, PUSH, ADD). By extracting and analysing opcode sequences, security researchers can detect malicious behaviors, obfuscation techniques, and unique signatures of malware. The first steps is reading the opcode unigram features from asm_opcode_unigram_40_features.csv,then each row contains frequencies of top 40 opcodes extracted from assembly code shown in Figure 1.4 Finally Merging the opcode features with trainLabels.csv to associate each sample with a class.

SERIAL NO	ID	SIZE	CLASS
0	0A32eTdBKayjCWhZqDOQ	3.509625	2
1	0ACDbR5M3ZhBJajygTuf	4.617676	7
2	0AguvpOCcaf2myVDYFGb	0.992920	8
3	0akIgwhWHYm1dzsNqBFx	5.087897	2
4	0aKlH1MRxLmv34QGhEJP	1.016846	8

Table 1.2 Malware Size and class

3.3 Preprocessing of Data

The preprocessing steps in the code involve multiple stages to prepare data for training an LSTM-based malware classification model. The first step is normalizing the data ,The byte frequency data is merged with the class labels. Table 1.2 consists of each feature (except ID and Class) is normalized using Min-Max Scaling. Table 1.3 explains the short brief for the types of Feature Normalization.

It improves the model convergence with gradient-based optimization performs better when features are on the similar scale and finally it enhances distance-Based models with algorithms like k-NN and K-Means clustering rely on Euclidean distance, which is affected by different feature scales. It helps with model convergence, especially for deep learning models. The second step is by splitting data into training and testing datasets. The dataset are split into X_train and X_test whereas maintaining the class distribution using stratified sampling. The third step is by handling Class Imbalance and Data Balancing using SMOTE technology. SMOTE is a widely used technique for addressing class imbalance in datasets. It helps improve model performance by generating synthetic samples for the minority class instead of simply duplicating existing and it identifies the minority class in an imbalanced dataset. Then selects the k-nearest neighbors for each sample in the minority class. It generates synthetic data points along the line between the original data point and its neighbors. At last it balances the dataset by increasing the sample count in the minority class. Data are again splitted to X_train and X_test, followed by SMOTE resampling finally the dataset is reshaped to align with the LSTM model format: (samples, timesteps, features) where timesteps is the number of extracted features.

Normalization Type	When to Use
Min-Max Scaling	When feature ranges are known and no extreme outliers exist.
Z-Score (Standardization)	When data follows a normal distribution.
Log Transformation	When data is skewed with large variations.
L2 Normalization	When working with high-dimensional sparse data (e.g., NLP, TF-IDF).

Table 1.3 Types of Feature Normalization

3.4 LSTM Classification and analysis

LSTM is a specialized form of Recurrent Neural Network designed to handle sequential data while effectively overcoming the vanishing the problem .It is widely used in time-series forecasting, NLP, and malware classification where sequential dependencies exist. An LSTM model is composed of several layers that process the input data step by step. The First layer is Input Layer which is used to defines the structure of input data. The Second layer is LSTM Layer which captured the sequential patterns in the data. The third layer is dense layers which is fully connected layers responsible for classifying data. Each layer plays a vital role in malware classification by leveraging opcode and byte sequence features.

3.4.1 Building the LSTM model

The LSTM layer is essential in the model's architecture, helping it learn patterns in sequential data LSTM Cell Architecture. The dataset is reshaped into **3D format** ([samples, timesteps, features]), required for LSTM models. Each LSTM cell comprise four main components

: the Forget gate, which decides what information to be discarded; the Input gate, which determines selects and incorporates new information; the memory state update, which refreshes the memory with essential data and the output control gate,which generates the specific final result.

3.4.2 Training & Testing

The training dataset consists of data used to fit and train the model to recognize patterns and learn the features. This dataset is fully accessible to the model throughout training. The test data collection, on the other hand, is a separate subset utilized to assess the model's performance, ensuring an accurate assessment of its effectiveness on unseen data. Table 1.4 explains clearly about the LSTM hyper parameter. By training the LSTM model, the categorical cross-entropy loss function and Adam optimizer are utilized to optimize performance. Categorical cross-entropy is widely used the loss function for multi-class classification, and it encourages the model to assign higher probabilities to the correct class labels. It is particularly effective when combined with SoftMax activation in the output layer. Adam Optimizer (Adaptive Moment Estimation) is which Combines Momentum & RMSprop that Uses both past gradients (momentum) and adaptive learning rates for efficient training. It faster Convergence ,adapts learning rate dynamically for each weight and robust to Noisy Gradients which works well with high-dimensional data like malware. To assess the model's performance the classification accuracy metric is utilized, which was trained for 300 epochs with a batch size of 32.

4. Result and Discussion

4.1 Evaluation Metrics

It evaluates the LSTM-based malware classification model using several metrics. Here's how the evaluation is done by test loss and accuracy Set. Once the training is done, the model will be evaluated on the testing using loss, which quantifies how it is predicted class probabilities align with the true class labels. Accuracy is which produced the percentage of correctly classified samples. This provides an overall measure of model performance. This model makes the predictions on the training data by probability distributions over the classes. np.argmax() converts probability distributions into discrete class labels by selecting the highest probability.

False Positive Rate(FPR) Metric: It evaluates the ratio of compromised states that are incorrectly recognized as normal, providing an understanding of the system's misclassification rate. A lower value indicates the signals enhanced performance. It is determined using the formula FPR=FP/N where FP represents the count of incorrectly identified the positive cases, and N is the overall count of abnormal vectors [9].

$$FPR = \frac{FP}{N} \quad (6)$$

False Rejection Rate(FFR) Metric: It measures the frequency of intrusion detection system incorrectly identifies the legitimate states as intrusions, resulting in false alarms that limit network access for authorized users. It is computed using the formula FRR=FNP , where FN denotes the count of incorrectly identified negative cases, and P is the overall count of positive data [9].

$$FRR = \frac{FN}{P} (7)$$

Accuracy Metric: This metric indicates the system's accuracy in accurately the identifying the both intrusion and non intrusion states. It is computed using the formula ACC=true results/total records ACC - \frac{\text results}}{\text{total records}} ACC=true result all record, where higher values indicates the better performance[9].

$$ACC = \frac{true\ result}{all\ record} \quad (8)$$

Precision Metric: Precision is an essential performance measure for assessing the accuracy of intrusion detection algorithms derived using the formula Precision=TP/TP+FP, it indicates how accuracy the system distinguishes real intrusions from all detected threats. A higher precision values signifies a lower occurrence of false positive [9].

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

In this scenario, TP denotes the number of positive cases correctly detected, Meanwhile, FP refers to the count of instances incorrectly classified as positive.

Recall Metric: It is determined using the following formula [9]:

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

Hyper parameter	Description	Recommended values
LSTM units	Number of memory cells	32, 64, 128,256
Return sequences	Whether tom return sequence outputs	True(multiple LSTM layers), false(final LSTM)
Batch size	Samples per training batch	16,32,64
Dropout Rate	Prevents overfitting	0.2-0.5
Activation	Function for neuron activation	ReLU(hidden layers), Softmax (output layer)
Optimizer	Controls learning rate adjustments	Adam(0.001)
Loss function	Measures error	Categorical Cross entropy
Epochs	Number of training cycles	20-100(with early stopping)

Table 1.4 LSTM hyper parameter with description and values

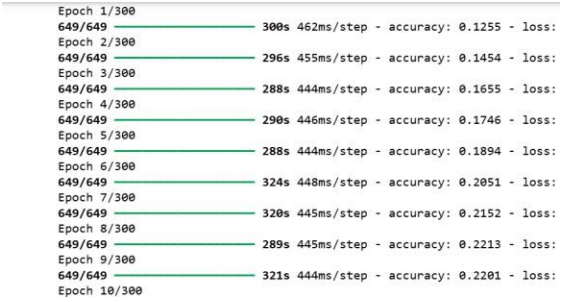


Figure 1.5 LSTM hyper parameter with batch size

S.No	Precision	Recall	F1 Score	Support
1	0.98	0.95	0.96	1233
2	0.97	0.97	0.97	1982
3	0.93	0.99	0.96	2354
4	0.99	0.95	0.97	380
5	1.00	0.91	0.95	34
6	0.98	0.94	0.96	601
7	1.00	0.94	0.97	318
8	0.99	0.97	0.98	982
9	0.99	0.96	0.97	810

Table 1.5 classification report

4.2 Performance Metrics

Precision evaluates the accuracy of positive predictions. Recall measures the proportion of actual positives correctly identified.F1 Score is used to balance precision and recall through their harmonic mean. Support is used to calculate the number of instances per class, Hyperparameter tuning entails selecting key parameters such as the number of LSTM units, learning rate, batch size and epochs. The process starts by defining a range of possible values for each hyperparameter. A population (swarm) of particles is then randomly initialized with positions and velocities within the search space. The fitness of each particle is assessed based on the validation accuracy of the LSTM model.

Train the LSTM model on the training data using the optimal hyper parameters. Then, assess its performance Here, **TP signifies the number of positive cases accurately identified**, While FN represents the number of cases incorrectly identified as negative.

F-measure Metric: This metric is derived using the following formula (11), serves as a balance between recall and precision [9]:

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

on the test data evaluation through key metrics like accuracy, precision, recall, and F1-score

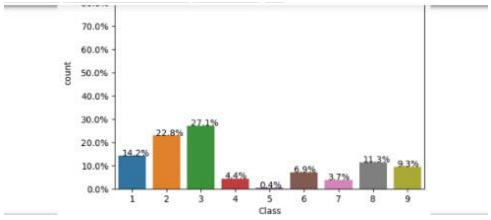
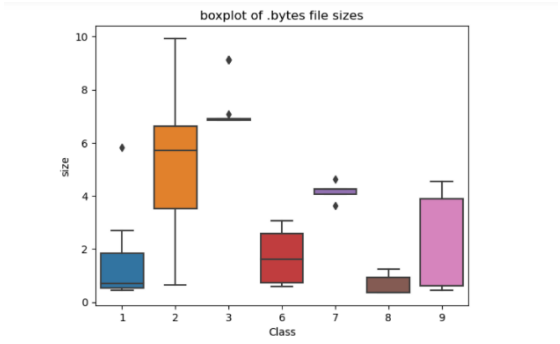
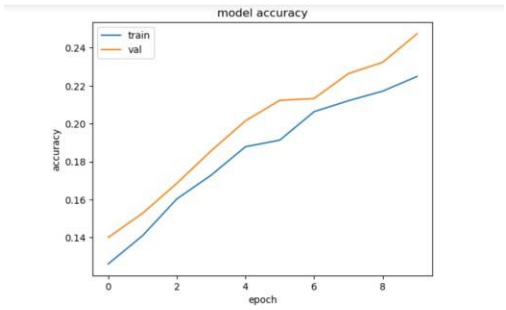


Figure 1.6 explains the total count and Malware Classification



This Figure 1.7 showcase about the Size and class of Malware



This Figure 1.8 showcase about the accuracy

A confusion matrix is utilized to evaluate classification performance across all malware categories in Figure 1.9

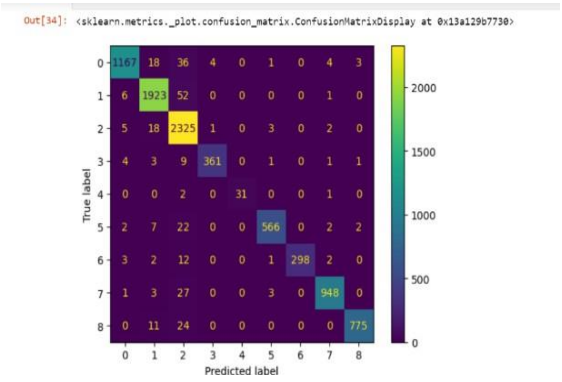


Figure 1.9 Confusion Matrix for Predicated Label

5. Conclusion

The proposed system effectively leverages deep learning techniques, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to enhance the detection and classification of IoT malware. Then by automatically deriving insights from raw data, opcode sequences and their entropy values, the system minimizes the need for human intervention in feature extraction, thus improving scalability and adaptability. The inclusion of Long Short-Term Memory (LSTM) networks further enhances the model's ability to handle long-range dependencies, ensuring more accurate malware detection. The integration of CNNs and RNNs offers a robust solution for IoT malware analysis, addressing challenges such as feature extraction, gradient issues, and detection accuracy, making it an important advancement in cybersecurity.

A key contribution of this research is the use of **LSTM networks** to handle the sequential nature of opcode and bytecode data. LSTM networks enable the model to capture long-term dependencies and sequential patterns in malware behavior, enhancing its ability to distinguish between various types of malware. The integration of **CNNs** further strengthens the model by efficiently extracting local features from the malware binary data, which is crucial for differentiating between malware families.

6. Future Work

This future work presents a parameter-efficient fine-tuning approach for malware detection using a pre-trained GPT-2 model. The system leverages Low-Rank Adaptation (LoRA) to enhance GPT-2 for binary sequence classification while reducing the number of trainable parameters. The GPT-2 tokenizer processes opcode sequences extracted from executables, mapping them into a structured format for classification. The dataset, provided in CSV format, is pre-processed and tokenized before being used for training. The Trainer API from Hugging Face facilitates fine-tuning, incorporating epoch-based evaluation and checkpoint saving. The trained model is then used for inference, classifying opcode sequences as malware or benign based on their patterns. This implementation demonstrates how transformer-based language models can be efficiently adapted for cybersecurity applications with minimal computational overhead.

By incorporating pre-trained transformer models such as GPT-2, adapted via LoRA, this future work aims to push the boundaries of modern malware detection systems.

The fusion of **natural language processing (NLP)** and **cybersecurity** presents a promising frontier, where models can learn semantic and syntactic relationships in malware code similarly to how they understand human language. The result is a scalable, interpretable, and high-performing system designed to meet the demands of evolving threats in **IoT and beyond**.

References

[1] Ngo, Quoc-Dung, Huy-Trung Nguyen, Van-Hoang Le, and Doan-Hieu Nguyen. "A survey of IoT malware and detection methods based on static features." *ICT express* 6, no. 4 (2020): 280-286.

[2] Jiang, Xingbin, Michele Lora, and Sudipta Chattopadhyay. "An experimental analysis of security vulnerabilities in industrial IoT devices." *ACM Transactions on Internet Technology (TOIT)* 20, no. 2 (2020): 1-24.

[3] Albulayhi, Khalid, Qasem Abu Al-Haija, Suliman A. Alsuhibany, Ananth A. Jillepalli, Mohammad Ashrafuzzaman, and Frederick T. Sheldon. "IoT intrusion detection using machine learning with a novel high performing feature selection method." *Applied Sciences* 12, no. 10 (2022): 5015.

[4] Paik, Joon-Young, Rize Jin, and Eun-Sun Cho. "Malware classification using a byte-granularity feature based on structural entropy." *Computational Intelligence* 38,no. 4 (2022): 1536-1558.

[5] Moon, Sunghyun, Youngho Kim, Hyunjong Lee, Donghoon Kim, and Doosung Hwang. "Evolved IoT malware detection using opcode category sequence through machine learning." In *2022 International Conference on Computer Communications and Networks (ICCCN)*, pp. 1-7. IEEE, 2022.

[6] Nazir, Ahsan, Jingsha He, Nafei Zhu, Saima Siraj Qureshi, Siraj Uddin Qureshi, Faheem Ullah, Ahsan Wajahat, and Muhammad Salman Pathan. "A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem." Ain Shams Engineering Journal 15, no. 7 (2024): 102777.

[7] Maran, Piragash, Timothy Tzen Vun Yap, Ji Jian Chin, Hu Ng, Vik Tor Goh, and Thiam Yong Kuek. "Comparison of machine learning models for IoT malware classification." In International Conference on Computer, Information Technology and Intelligent Computing (CITIC 2022), pp. 15-28. Atlantis Press, 2022.

[8] Riaz, Sharjeel, Shahzad Latif, Syed Muhammad Usman, Syed Sajid Ullah, Abeer D. Algarni, Amanullah Yasin, Aamir Anwar, Hela Elmannai, and Saddam Hussain. "Malware detection in internet of things (IoT) devices using deep learning." *Sensors* 22, no. 23 (2022): 9305.

[9] Mehrban, Ali, and Pegah Ahadian. "Malware detection in iot systems using machine learning techniques." *arXiv preprint arXiv:2312.17683* (2023).

[10] Akhtar, Muhammad Shoaib, and Tao Feng. "Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time." *Symmetry* 14, no. 11 (2022): 2308.

[11] Ngo, Quoc-Dung, Huy-Trung Nguyen, Van-Hoang Le, and Doan-Hieu Nguyen. "A survey of IoT malware and detection methods based on static features." *ICT express* 6, no. 4 (2020): 280-286.

[12] Anand, Abhishek, Jyoti Prakash Singh, Rida Sohail Khan, Anjali Kumari, and Divya Mishra. "Android Malware Detection using LSTM with Smali Codes." (2023)

[13] Ali, Muhammad Mumtaz, Faiqa Maqsood, Weiyan Hou, Zhenfei Wang, Khizar Hameed, and Qasim Zia. "Machine Learning- Based Malware Detection for IoT Devices: Understanding the Evolving Threat Landscape and Strategies for Protection." (2023)

[14] Gad, Abdallah R., Mohamed Haggag, Ahmed A. Nashat, and Tamer M. Barakat. "A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset." International Journal of Advanced Computer Science and Applications 13, no. 6 (2022).

[15] Li, Shudong, Qianqing Zhang, Xiaobo Wu, Weihong Han, and Zhihong Tian. "Attribution classification method of APT malware in IoT using machine learning techniques." Security and Communication Networks 2021, no. 1 (2021): 9396141

[16] Darabian, Hamid, Ali Dehghantanha, Sattar Hashemi, Sajad Homayoun, and Kim-Kwang Raymond Choo. "An opcode-based technique for polymorphic Internet of Things malware detection." Concurrency and Computation: Practice and Experience 32, no. 6 (2020): e5173.