Efficient method for Android Malware Family Detection and Classification using Sequential Network with ECLAT

Ayush Wardhan sahu[1], Prof. Satendra Sonare[2]

[1]Reseacrh Scholar, Department of CSE, Gyan Ganga Institute of Technology and sciences, Jabalpur, M.P.
[2]Professor, Departmet of CSE, Gyan Ganga Institute of Technology and Sciences, Jabalpur, M.P.

---***---

Abstract – Android's popularity makes it a lucrative target for cybercriminals. Malware with different behavior patterns that specifically target user routines is constantly entering the market. For this reason, knowing how to identify different forms of malware is essential to anti-malware protection. Android malware has become a serious threat to our daily lives, so it is urgently necessary to effectively mitigate or defend against it. Recently, many Android malware analysis approaches and tools have been proposed to protect legitimate users from the threat. However, most approaches focus on malware detection, while only a few consider malware classification or malware characterization. This paper proposes the use of ECLAT-based machine learning and deep learning methods to classify malware families and categories based on many different datasets in order to evaluate and select appropriate methods for each dataset.

Keywords: Android malware, Malware Features, Deep Learning, ECLAT, Sequential Neural Network, Accuracy.

1. INTRODUCTION

To protect computer systems, we need to detect malware as soon as it infects systems. Malware detection is the process of analyzing a suspicious file and identifying whether it is malware or benign. Malware classification is a step further. After a file is identified as malware, by entering a category or group of malware known as a malware classification. Malware detection requires 3-step operations:

1. Malware files are analyzed using appropriate tools. 2. Static and dynamic elements are extracted from the analyzed files.

3. Functions are grouped in certain ways to separate malicious software from benign software. Various sciences and techniques including data science, machine learning, and heuristics, as well as technologies such as cloud computing, big data, and blockchain, are used in these processes to increase detection rates. There are different approaches to detect malware using the above techniques and technologies. These approaches are mainly signature, behavior, model checking and heuristic detection [1], [2]. The names of these approaches vary according to the techniques and technologies used. The signature-based approach is effective for known and similar versions of the same malware. However, it failed to detect previously unseen malware. Although detection approaches based on behavior, heuristics, and model checking are effective in detecting some parts of unknown malware, they cannot show the same performance in detecting complex malware variants that use obfuscation and packaging techniques. A deep learning approach is starting to be used as a new paradigm to overcome the shortcomings of existing malware detection and classification approaches. Deep learning has been widely used in various fields including image processing, computer vision, human activity recognition [3], driving safety [4], facial emotion recognition [5], and natural language processing. However, it is little used in the field of cyber security, especially in malware detection. Deep learning is a subset of artificial intelligence that operates on artificial neural networks (ANNs). Deep learning uses multiple hidden layers and learns from examples. Recently, several deep learning architectures such as deep neural networks (DNNs), deep belief networks (DBNs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs) have been used to improve model performance. Deep learning brings many advantages over the traditional learning scheme:

1. The DL model can automatically generate high-level elements from existing elements.

2. DL reduces the need for feature engineering.

3. DL can handle unstructured data efficiently.

4. DL can handle very large datasets.

5. DL reduces the element space.

6. DL can effectively perform unsupervised, semi-supervised and supervised learning.

7. DL reduces costs and increases accuracy.

1.2 Types of Malware

Malware, Short For Malicious Software, Is Software Designed by Cyber Attackers to Access or Harm a Computer or Network without the Victim's Knowledge. Malware Is Defined As Any Software Designed To Cause Direct Harm. Despite A 39% Decrease In Global Malware Volume In 2020, Malware Attacks Continue. At The Same Time, Some Types

Of Malware Are Linked To The Use Of Similar Attacks, Such As The Use Of Explosives, Meaning That Some Ideas Are Created By The Victims Themselves, Phishing And Social Engineering Tactics Are Used To Distribute Malware Directly To Victims; ; Or Mobile Malware, Which Is Malware That Targets Mobile Devices. The Most Common Types of Malware Are:

1. Malware viruses [6]

Malware Is A Type Of Malicious Software, Usually In The Form Of Code That Is Added To An Application, Program, Or System And Distributed By Its Victims. Most Types Of Malware Are Similar To Viruses In That They Require A Host (Such As A Device) To Survive. They Remain Active Until An Attack Occurs, Probably Caused By A User Downloading An Email Link (Usually An .Exe File, Meaning An "Executable" File). Documents Are Copied From One Computer To Another, Causing the Most Damage. Finally, malware can:

- Capture applications.
- Send infected files to individuals
- Steal data
- Eliminate DDoS attacks
- Attack ransomware software attacks.

Example: iloveyou virus, 2000.

2. Worm malware [6]

Ultimately, worm malware can:

- Delete or modify files
- Steal data
- Install backdoors for hackers
- Launch DDoS attacks
- Launch ransomware attacks
- Create botnets
- Infect many computers at once.

Examples: Sql slammer, 2003: considered one of the fastest growing malware ever, sql slammer exploits vulnerability in Microsoft sql server software. The attack lasted only 10 minutes and affected thousands of servers.

3. Trojan malware [6]

A trojan horse is a type of malware that disguises itself as real software, applications, or files, causing users to download and lose control of their devices. Once installed, trojans can do what they were designed to do, such as damage, destroy, steal, or cause other problems with your data or network. Download websites or direct messages from email links. Like viruses, these must be done by the user. The difference when comparing malware and trojans is that viruses are host-dependent, while trojans are not. Trojans do not replicate themselves like viruses.

Ultimately, trojan malware can:

- Delete, modify, or steal data
- Spy on users
- Access networks
- Launch DDoS attacks
- Take remote control of devices.

Examples: Zeus/Zbot, 2011: This banking Trojan leveraged keystroke logging to steal credentials and also account balances.

4. Ransomware [6]: As the name suggests, ransomware is malware that carries a ransom. It locks and encrypts the victim's device or data and demands a ransom to regain access. This usually occurs when the victim has downloaded the malware through an email link or a link from an unknown source. Once installed, the malware can create a backdoor that allows hackers to access the device and then begin encrypting the data, thus locking the entire owner out of the device until they pay a ransom to regain ownership. It is worth noting that ransomware is increasingly being paid in crypto currency, sometimes called crypto-malware.

Ultimately, ransomware can:

- Hold devices hostage
- Make data inaccessible through encryption
- Result in financial loss

Examples: WannaCry, 2017: this ransomware attack targeted thousands of computers running the windows operating system worldwide and spread to corporate partners worldwide. Victims are asked to pay a bitcoin fee to get their data back.

5. Bots or botnets [6]

Other times, bots can act as "spiders," meaning the program is checking the network to see when hackers are using the site and using a security site, or if you prefer, a bot to do so. Malware. In some cases, botnets attack devices directly, allowing cybercriminals to control them remotely.

Finally, botnets or botnets can:

- Launch DDoS attacks
- Record activities including keystrokes, webcams, and screenshots
- Email through your device can send phishing emails
- Hackers can use device remotely

Examples: Mirai, 2016 : This botnet attack targeted Internet of Things devices and, from there, leveraged DDoS attacks.

6. Adware malware

As the name suggests, adware is advertising-related malware. Also known as adware, adware displays unwanted advertisements on your computer, sometimes in the form of pop-up ads that track the user's browsing experience. Where adware can go wrong is when these ads can distort your messages; download to spread.

Finally, adware can:

- be annoying
- redirect users to malicious websites
- install spyware
- share user information with us.

For example: fireball, 2017: this adware infected approximately 250 million devices by tracking victims' online gaming through browser hijacking.

7. Spyware [6]

As the name suggests, adware is advertising-related malware. Also known as adware, adware displays unwanted advertisements on your computer, sometimes in the form of pop-up ads that track the user's browsing experience. Where adware can go wrong is when these ads can distort your messages; download to spread. Finally, spyware can:

- invade a person's privacy
- collect confidential information, including recording keystrokes
- steal information
- cause theft or credit card fraud

Example: dark hotel, 2014: this key logging spyware targets government and corporate executives using hotel wi-fi.

8. Rootkits [6]

A rootkit is a type of malware that allows cybercriminals to take control of a victim's device, usually without the victim's knowledge. Because rootkits are designed to remain silent, they can steal or replace security software, allowing the malware to persist on your computer for long periods of time and cause serious damage, according to the report.

Finally, rootkits can:

- Control devices remotely
- provide cybercriminals with controlled access to devices
- Monitor user activity

Examples: zacinlo, 2012: this rootkit remained hidden until: first discovered around 2017, this virus often infects adware on windows devices and disables antivirus software.

9. Fileless malware [6]

A rootkit is a type of malware that allows cybercriminals to take control of a victim's device, usually without the victim's knowledge. Because rootkits are designed to remain silent, they can steal or replace security software, allowing the malware to persist on your computer for long periods of time and cause serious damage, according to the report.

Finally, Fileless malware can:

- break antivirus software
- steal data

Example: astaroth, 2019: this Fileless malware is a real data stealer and mostly attacks windows devices and targets specific countries including Brazil.

10. Malvertising [6]

Not to be confused with adware, adware is malware that comes from advertisements on legitimate websites. However, adware is a type of malware that is already present on the device. Both attacks rely on online advertisements to cause harm. You can fall victim to an ad virus by clicking on it (which cybercriminals may pay to have on the website), or you can fall victim to downloads by visiting a website with fake advertisements.

Finally, malvertising can:

- Lead to ransomware attacks
- Lead to data theft
- Lead to credit card fraud

Examples of malicious advertising: media, 2016: the New York Times, the BBC, aol and other unknown news sites target readers with the aim of hacking computers and demanding ransoms.

Figure 1.1: Types of Malwares [6].

1.4 Deployment of Malwares

Malware is often spread via email. According to statistics, 94% of them are sent via email. However, cybercriminals still use different methods to attack malware. These are just a few of their common ideas, some are mixed.

A man-in-the-browser attack occurs when an attacker places malware on a computer to collect information sent to victims and specific websites, and then installs the malware in the browser without the user's knowledge.

• Security exploitation is when cybercriminals manually search for vulnerabilities in devices and networks and then inject malware into them.

™ Electronic Devices are an alternative to exploiting vulnerabilities. These are pre-written scripts that search for vulnerabilities in devices and ultimately inject malware into that security system.

Drive-by downloading is when a user accesses a malicious website that hosts a malware attack. Phishing can include phishing or SMS scams.

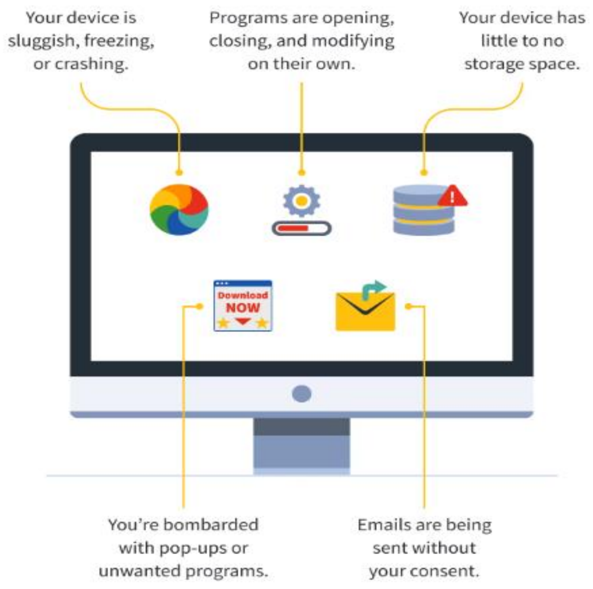

Your device is sluggish, freezing, or crashing.
Programs are opening, closing, and modifying on their own.
Your device has little to no storage space.
Download NOW
You're bombarded with pop-ups or unwanted programs.
Emails are being sent without your consent.

Figure 1.2: Symptoms of Malware attacks.

2. Literature Survey

2.1 Malware Detection and Classification: Malware detection and classification is a long process. Many techniques and methods are used in these stages. To identify malware, you must first use the relevant analysis tools. Secondly, the results of the device are closed and the features are manually extracted or cut. At this stage, the information is only used to get good results in the process. After that, the extraction process is selected according to certain rules. Finally, the selected features are trained by machine learning algorithms or formal learning techniques to distinguish bad software from good software. Malware identification, detection, and classification are shown in figure 2.1[7]. In order to better understand the content and purpose of malware, further classification can be done by determining the type and category of malware it belongs to. This section is divided into four sections: malware detection tools and platforms, malware analysis, malware feature extraction, and detection and classification.

1. Desktops and laptops.

2. Mobile devices

3. Internet of things devices.

4. Cloud computing platform. Other devices such as smartphones, personal assistants (pdas), and the internet of things (iot) have also become known over time. Computer viruses became very popular in the 1990s and 2000s [8]. From 2000 to 2010, the first computer viruses appeared, followed by Trojan horses. Ransomware has become a very popular malware since 2010 [9]. From 1990 to 2010, most types of malware were written for regular computers, as well as for the windows operating system (os), as well as other operating systems other than windows (such as unix, different versions of linux, and MacOS). Therefore, a malware detection method for computers is ready. However, after 2010, mobile devices such as smartphones, tablets, and pdas became popular. After that, cloud computing environments and iot devices became quite popular. According to recent research, the number of smartphones has surpassed computers, and the difference is increasing day by day. People use mobile apps more than the web version of the program. The number of iot devices is also increasing. As the use of smartphones and iot devices is more popular than regular computers, cybercriminals' attention has also shifted from regular computers to smartphones and iot devices. Cybercriminals modify existing malware and create different versions of the same malware that can run on these devices. According to mcafee's report, there has been an increase in banking trojans, backdoors, and fake apps targeting mobile platforms [10]. Also, malicious attacks involving crypto currencies, social media, iot devices, and cloud computing environments are increasing. These reasons are that malware detection is moving from computers to mobile iot devices and cloud environments. Cloud computing has many advantages for malware detection, including easy access, more power, and larger data [11]. Therefore, up-to-date information on malware detection and distribution for mobile phones and iot is collected using deep learning and used in the cloud environment.

2.2 Machine Learning-Based Detection

Data mining and machine learning techniques identify malware based on features obtained from static analysis, negative analysis, visualization-based analysis, or a combination of analyses. Data mining classifiers are trained using various features to classify unknown malware. Some studies use feature ranking [12] to perform effective malware classification using machine learning algorithms. The framework proposed by [13] is based on static features that can be applied to three learning algorithms, namely inductive student rule, probabilistic method, and multiple classifier systems. Their approach is limited to the world's largest models. Reference [14] uses machine learning techniques such as naive Bayes, decision trees, support vector machines, and n-gram feature-based boosting. The authors theorize that boosted decision trees are superior to other methods. Their approach is limited to crime detection and identifying malware execution characteristics. Static PE archive features were extracted by testing with executables [15] and machine learning classifiers (such as Naive Bayes, Bagged J48, KNN, multilayer perceptron (MLP), and J48 decision trees using entropy thresholding and unbagged examples for classification). Reference [16] constructed vectors from opcode sequences of executables and trained a machine learning classifier. Their approach is used only for packing. The phrase sequences are extracted from PE archives [17] and negative examples are created to represent them. The authors perform malware detection using each nearest neighbor (ANN) classifier. However, the success of machine learning-based measurement tools depends on the extracted features and the capabilities of the selected machine learning model. A classifier may fail if it cannot provide clues due to packing, obfuscation, and minor changes in the order of features made by the malware author. It is also used to identify and isolate malware to augment malware analysis methods for malware analysis. A combination of static and dynamic features improves detection [18], [19]. The implementation [20] combines static features such as long frequency vectors and print data vectors with dynamic API features into a single vector. It extracts opcode sequences and system call signatures from binary archives using dynamic analysis. These features [21], [22] are seen later in the image. A similar prediction method [22], [23] consists of static and dynamic features based on image matrix vector representation. The model is represented using a new method [24] by creating groups based on similar operations for each type of resource (e.g., File names, names, mutexes, and name register). They use machine learning classifiers to classify malware and cleanware. This technique is not capable of providing general visibility for malware detection because new malware and different types of existing malware are always present in the network. Automatic Android Malware Detection (AAMD-OELAC) technology integrates learning. The main goal of AAMD-OELAC technology is the automatic classification and identification of Android malware. To achieve this, the AAMD-OELAC system processes the data in a preliminary stage. For the Android malware detection process, the AAMD-OELAC technology based on the learning process using three ML models, namely Least Squares Support Vector Machine (LS-SVM), Kernel Extreme Learning Machine (KELM), and Regularized Random Vector Functional Connection Neural Network (RRVFLN). Finally, the av-av optimization (HPO) method is used to correct the disadvantages of our deep learning, which helps to improve the malware detection results. An experimental test was conducted to demonstrate the effectiveness of the AAMD-OELAC method. The simulation results demonstrate the superiority of the AAMD-OELAC technology over other existing machine learning-based antimalware solutions. The proposed system is based on hierarchical integration, which follows the basic features of deep learning but will be more efficient in the future. The proposed method does not require hyperparameters tuning or Backpropagation and can reduce model complexity. The proposed model outperforms other state-of-the-art methods, achieving 98.65%, 97.2%, and 97.43% detection rate for Malimg, BIG 2015, and MaleVis malware datasets, respectively. The results show that the solution is able to detect new and advanced malware due to its different capabilities. Find framework. The framework uses multiple regression methods. Authorization is one of the most important aspects of Android OS security; application authorization is extracted from static analysis, and application security analysis is performed through machine learning. According to different horizontal malware types, permissions are requested from two distributions based on Android malware detection. These classifications are compared with simple machine learning methods such as support vector machine, k-nearest neighbors, naive Bayes, and decision trees on four different datasets. In addition, bagging, one of the hybrid learning methods, is used to generate different distributions to improve the classification efficiency. Therefore, classification algorithms based on linear regression models can achieve high performance without the need for very complex classification algorithms. Engineering, decompiling, or disassembling the evaluation application in a way that violates the application license and terms of use. Moreover, these solutions often use the same machine learning (ML) model to identify different types of malware, which leads to many vulnerabilities. In this context, we propose a method for detecting Android malware that includes a set of dedicated detection tools, all based on rules and ML techniques, at the same level. Different application cycles (Pre- and Post-) Perform several analysis phases. Our method [25] also differs from the case solutions in that it is not affected by using a method that does not violate the application form and terms to obtain the application function.

3. Proposed Work

The proposed model will have two stages: analysis and deployment stage. During the analysis, each signature is checked for malware and data quality. Both static and dynamic features will be considered. For better results, the number of features will increase to 215. The second stage involves classifying malware into similar categories based on their static and dynamic features. In this stage, deep learning is used for malware detection and classification. The scanning stage, inspired by the concept of convolutional neural networks (CNN), will be used to preserve the relationship between the original pixels. Each profile in the training process is an input-level classification. The first stage starts with the analysis of input data from parallel processing. The following diagram shows the proposed working model.

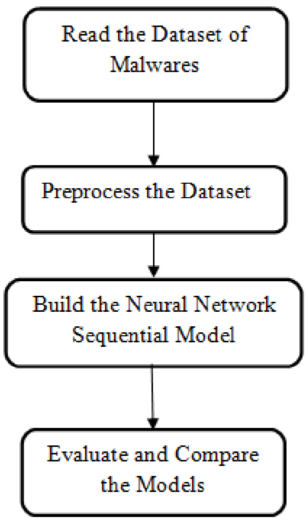

Read the Dataset of Malwares
Preprocess the Dataset
Build the Neural Network Sequential Model
Evaluate and Compare the Models

Figure 4.1: Proposed model steps.

4.5 Feature Selection and Dimensionality Reduction

Feature Selection and Dimensionality Reduction are important methods to improve model performance in machine learning by removing unnecessary features and focusing on data features, reducing overfitting to high-dimensional datasets and improving performance. They also contribute to model interpretation, solving multicollinearity problems, handling cases with little or no data, and improving model capabilities by making model construction easier. These ideas also help create good and effective beautiful designs by guiding the architecture and helping to understand how features affect different targets. ECLAT was selected for special selection in Android malware detection due to its efficiency and effectiveness in the problem. These techniques are known to improve model performance and interpretation and are frequently used in machine learning, making them a suitable choice for this study.

A. Training-Test Split

In machine learning and data science, a technique called training-test split is used to evaluate the performance of a model. Training-test split is a process that splits the existing data into two groups: a training sample and a test sample. A percentage of the data is split between training and testing. A ratio provides 70-80% of the material for the training process and the remaining 20-30% for testing. But the percentage will vary. Our model's data is divided into two parts: training and testing. 90% Of the dataset is used for training and 10% for testing. These are two new record stores with a total of 41 entries. The final result is the answer, which has two possible answers: 1 or 0. The first 40 items are features, the last items are responses. - Train size = 13528 - test size = 1503. Model Design

The proposed model is designed to study classification problems using small data, which is a special type of neural in our work. The network is usually used as a link. As a result, the implementation of the planning function has proven to be very effective because it aims to study the similarities rather than statistics between two proposals.

C. Model evaluation

After training, the network can compare two data samples. The first query sample uses the training network embeddings and then compares using the same ECLAT algorithm. Use distance scores to generate binary options. Training models

Design models commonly used to solve Android malware problems were used to train our models. Using a set of points as input, the model learns to predict whether two data points belong to the same category. The two-layer structure is the first part of the design, followed by the layer structure, distance layer and sigmoid release layer, each with 64 hidden holes. The model using stochastic gradient descent is optimized after training with binary cross-entropy loss. Training was conducted in 10 sessions with a group of 8 people. In addition, a function generator is used in this study to generate data pairs for each dataset; 80% of the data is used for training and 20% for use. To ensure that the model does not conform to training materials, training and misuse, and is followed during training. Two effective methods are used to reduce the effects of class inequality during model training: stratified sampling and class weighting.

Malware detection often faces the problem of class inequality, where the number of examples in one class greatly exceeds the number of examples in another class. By using undersampling, you ensure that each mini-batch used for training has a proportional representation of each class, thus controlling the class distribution of the original data. This strategy ensures that the minority group receives sufficient attention during training and prevents the model from biasing towards the dominant group. Weighting units are also included in the optimization process. This model focuses more on analyzing data from a small class by assigning higher weights to fewer classes, thus solving the problem of under-classification. This process improves the model’s ability to perform and correct classification of most and least classes, making Android malware detection more efficient and reliable. Training and testing accuracy should be taken into account in a full evaluation of machine learning models. It has many features that will help increase the usability and reliability of your model. First, it helps identify over-fitting, which occurs when a model focuses too much on memorizing information and underperforms when presented with new information. Overfitting can be detected by comparing the accuracy of the training and test data, and appropriate measures can be taken to correct it.

A model accuracy rate above 99% indicates that the model is able to predict patterns well in the test data. However, the model will become training data, which will cause performance to degrade on new data. Validation drops and accuracy are monitored during training to prevent overfitting. Monitoring redundancy is important because it provides information about the model's ability to generalize and integrate. In Figure 6.42, both the training and validation losses show a decreasing trend indicating that the weights of the model are optimized and the model is learning new information from the data sheet. This model successfully reduces the binary cross entropy loss of the training data as can be seen from the decrease in learning loss and this also shows how good the model is for the study materials. This is supported by the fact that the validation loss also decreases as the model generalizes to the validation data.

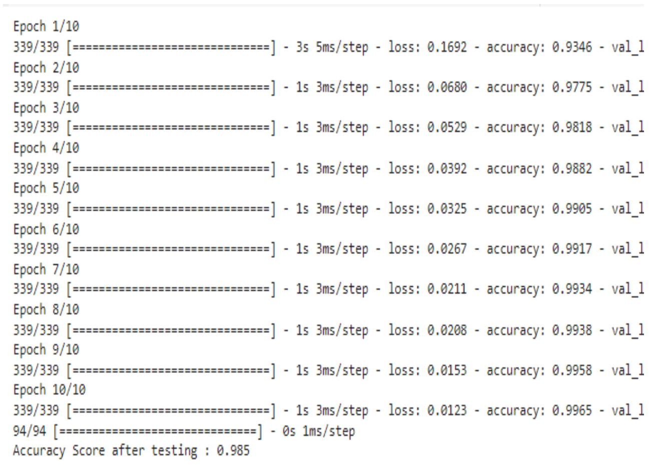
4. Result Analysis

The data we use in our paper is the well-known Drebin malware dataset. To use this data, we need to select individual features and then specify our model around them. To find the best features for our use case, we use a selection process to search for the most important features that detect malware.

The list of Android apps classified as dangerous or malicious is publicly available and is called the Drebin dataset. The data was generated from a 2014 study by researchers at Northeastern University to identify malware on Android devices [37]. The researchers separated each program in the dataset into bad and not-so-bad projects using static and dynamic analysis. The database contains 5,560 apps from 179 different malware families. The collection also includes 13,106 samples from other sources, including Android websites, malware forums, and security blogs, as well as 96,150 apps from the Google Play Store, 19,545 apps from various other stores in China, 2,810 apps from other Russian app stores, and more. The five main stages of its implementation are preprocessing, data partitioning, model architecture, training, and evaluation. The researchers used two methods to find the suspected malware, and then analyzed the results. The Drebin dataset has become a favorite of mobile security researchers in many studies and research initiatives. The document is available for public download from the Northeastern University website.

Chart.

After pre-processing, count the occurrence of each class. For counting, first separate the features and labels. Calculate the majority and minority class labels. Separate the majority and minority class. The results are shown below I figures.



```
Epoch 1/10
339/339 [==============================] - 3s 5ms/step - loss: 0.1692 - accuracy: 0.9346 - val_l
Epoch 2/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0680 - accuracy: 0.9775 - val_l
Epoch 3/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0529 - accuracy: 0.9818 - val_l
Epoch 4/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0392 - accuracy: 0.9882 - val_l
Epoch 5/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0325 - accuracy: 0.9905 - val_l
Epoch 6/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0267 - accuracy: 0.9917 - val_l
Epoch 7/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0211 - accuracy: 0.9934 - val_l
Epoch 8/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0208 - accuracy: 0.9938 - val_l
Epoch 9/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0153 - accuracy: 0.9958 - val_l
Epoch 10/10
339/339 [==============================] - 1s 3ms/step - loss: 0.0123 - accuracy: 0.9965 - val_l
94/94 [==============================] - 0s 1ms/step
Accuracy Score after testing : 0.985
```

Figure 4.1: Results of CNN after 10 epochs.

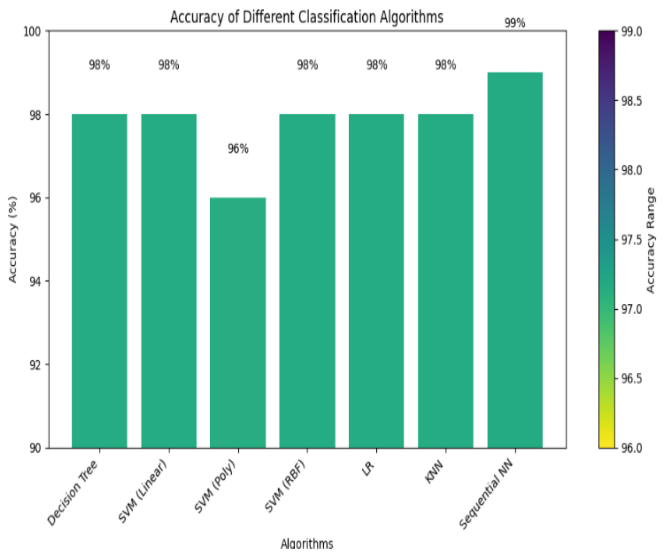


Figure 4.2: Accuracy comparison of all implemented Classifiers.

5. Conclusion

In summary, the study, class weights and stratified sampling are used during model optimization to address issues with class imbalance that are frequently present in malware detection. The effectiveness of the model is thoroughly assessed, accounting for both testing and training accuracy. Notably, the model outperforms earlier techniques like Logistic Regression, random forest classifier, KNN, XGBoost by achieving an impressive accuracy rate of 99% on the Drebin dataset and other dataset. The performance of the model is fully analyzed using a set of evaluation metrics, including accuracy, recall, precision, F1 score, and confusion matrix.

6. References

[1] O. Aslan and R. Samet, "A comprehensive review on malware detection approaches," IEEE Access, vol. 8, pp. 6249–6271, Jan. 2020.

[2] R. Komatwar and M. Kokare, "A survey on malware detection and classification," J. Appl. Secur. Res., pp. 1–31, Aug. 2020.

[3] A. A. Yilmaz, M. S. Guzel, E. Bostanci, and I. Askerzade, "A novel action recognition framework based on deep-learning and genetic algorithms," IEEE Access, vol. 8, pp. 100631–100644, 2020.

[4] A. A. Yilmaz, M. S. Guzel, I. Askerbeyli, and E. Bostanci, "A vehicle detection approach using deep learning methodologies," in Proc. Int. Conf. Theor. Appl. Comput. Sci. Eng., Nov. 2018, pp. 64–71.

[5] A. A. Yılmaz, M. S. Guzel, I. Askerbeyli, and E. Bostancı, "A hybrid facial emotion recognition framework using deep learning methodologies," in Human-Computer Interaction, T. Ozseven, Eds. Hauppauge, NY, USA: Nova Science Publishers, 2020.

[6] https://us.norton.com/blog/malware/types-of-malware.

[7] O. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[8] J. Love. (2018). A Brief History of Malware—Its Evolution and Impact. Accessed: Mar. 20, 2021. [Online]. Available: https://www.lastline. Com/blog/history-of-malware-its's-evolution-and-impact.

[9] D. Palmer. (2017). Ransomware: Security Researchers Spot Emerging New Strain of Malware. Accessed: Mar. 20, 2021. [Online]. Available: https://www.zdnet.com/article/ransomware-security-researchers-spotemerging-new-strain-of-malware.

[10] (2020). McAfee Mobile Threat Report Q1. Accessed: Mar. 21, 2021. [Online]. Available: https://www.mcafee.com/content/dam/consumer/enus/docs/2020-Mobile-Threat-Report.pdf.

[11] O. Aslan, M. Ozkan-Okay, and D. Gupta, "A review of cloud-based malware detection system: Opportunities, advances and challenges," Eur. J. Eng. Technol. Res., vol. 6, no. 3, pp. 1–8, Mar. 2021.

[12] M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," Expert Syst. Appl., vol. 143, Apr. 2020, Art. no. 113022, doi: 10.1016/j.eswa.2019.113022.

[13] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in Proc. IEEE Symp. Secur. Privacy. (S&P), May 2000, pp. 38–49.

[14] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2004, pp. 470–478.

[15] R. Perdisci, A. Lanzi, and W. Lee, "Classification of packed executables for accurate computer virus detection," Pattern Recognit. Lett, vol. 29, no. 14, pp. 1941–1946, Oct. 2008, doi: 10.1016/j.patrec.2008.06.016.

[16] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," Inf. Sci., vol. 231, pp. 64–82, May 2013, doi: 10.1016/j.ins.2011.08.020.

[17] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," Expert Syst. Appl., vol. 52, pp. 16–25, Jun. 2016, doi: 10.1016/j.eswa.2016.01.002.

[18] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2004, pp. 470–478.

[19] H.-J. Li, Z. Wang, J. Pei, J. Cao, and Y. Shi, "Optimal estimation of low-rank factors via feature level data fusion of multiplex signal systems," IEEE Trans. Knowl. Data Eng., early access, Aug. 13, 2020, doi: 10.1109/TKDE.2020.3015914.

[20] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," J. Netw. Comput. Appl., vol. 36, no. 2, pp. 646–656, Mar. 2013, doi: 10.1016/j.jnca.2012.10.004.

[21] J. Zhang, Z. Qin, H. Yin, L. Ou, S. Xiao, and Y. Hu, "Malware variant detection using opcode image recognition with small training sets," in Proc. 25th Int. Conf. Comput. Commun. Netw. (ICCCN), Aug. 2016, pp. 1–9.

[22] K. Han, B. Kang, and E. G. Im, "Malware analysis using visualized image matrices," Sci. World J., vol. 2014, pp. 1–15, Jul. 2014, doi: 10.1155/2014/132713.

[23] H.-J. Li, L. Wang, Y. Zhang, and M. Perc, "Optimization of identifiability for efficient community detection," New J. Phys., vol. 22, no. 6, Jun. 2020, Art. no. 063035, doi: 10.1088/1367-2630/ab8e5e.

[24] J. Stiborek, T. Pevný, and M. Rehák, "multiple instances learning for malware classification," Expert Syst. Appl., vol. 93, pp. 346–357, Mar. 2018, doi: 10.1016/j.eswa.2017.10.036.

[25] L. d. Costa and V. Moia, "A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques," in IEEE Access, vol. 11, pp. 73127-73144, 2023, doi: 10.1109/ACCESS.2023.3296606.