Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 45 No. 2 (2024)

Analysis of Malware Detection in Downloaded Files Based on CNN-SVM-RBF Approach

Mohamed Uvaze Ahamed Ayoobkhan [1],Thammisetty Swetha[2],Neerav Nishant [3] ,Rashmi Shekhar [4],V Janakiraman [5],Vishal Ratansing Patil [6]

[1] *Department of Computer Science, New Uzbekistan University, Tashkent, Uzbekistan,*

[2] *Department of CSE-Data Science, Madanapalle, Institute of Technology & Science, Manadanapalle, India,*

[3] *Department of Computer Science and Engineering, School of Engineering, Babu Banarasi Das University, Lucknow, India,*

[4] *Amity Institute of Information Technology, Amity University Patna, Rupaspur Patna, India,*

[5] *Department of Mechanical Engineering, Prince Shri Venkateshwara Padmavathy Engineering College, Chennai, Tamil Nadu,*

[6]*Department of CSE, AIML, Pimpri Chinchwad College of Engineering Nigdi, Pune, India,*

Abstract. It is possible for malicious coders to take advantage of programming errors committed by the developers themselves. These blunders can be the result of sloppy coding or logical flaws in the original design. Low-level languages like C and C++ make it possible to use pointers, which allow for arbitrary memory access within a program's address space. Unwary programmers' misuse of these features can result in illegal memory reads and writes. Malicious programmers use these unapproved reads and writes to cause even more devastation in the system. The proposed procedure begins with three stages: preprocessing, feature extraction, and model training. Normalization and regularization of data are part of the preprocessing phase. In order to pick features, extract N-gram features, and utilize Windows API functions. After features have been selected, the models are trained with CNN-RBF-SVM. The proposed method outperforms the two most used alternatives, RBF-SVM and CNN.

Keywords: *Convolutional Neural Network (CNN) · Support Vector Machine (SVM) · Internet of Things (IoT).*

1 Introduction

Malware detection still frequently makes use of tried-and-true methods including "signature," "resources," and "components". In addition, modern anti-virus software and web-based malware scanners aren't up to snuff when it comes to dealing with the constantly-evolving malware threats and novel malware signatures. Again, smartphones don't have the horsepower to constantly scan for spyware. Therefore, a more efficient method that can exceed the capabilities of current systems is required for detecting the most cutting-edge malware. Machine learning is now crucial in Malware Detection. Malware detection using supervised machine learning classifiers requires a dataset. Machine Learning models need to be trained on data. Machine learning classifiers are essential for malware detection because they can automatically acquire knowledge from training data and create classifications accordingly. We found its difficulty to process huge files, high resource utilization, and failure to recognize newly released malware using a static study utilizing conventional methodologies and internet scanners. As a result, malware detection requires the use of machine learning classifiers. In this study, we present a strategy for malware detection based on the training and assessment of supervised machine learning classifiers. To research malware detection for a wide variety of viruses, we compiled a dataset utilizing malware files from the world's top malware projects. Any business that uses the Internet must prioritize data protection. The ransom ware attack

model has shown to be quite efficient in recent years. One of the most widespread types of malware today is ransom ware. Ransom ware has had a devastating effect on many people and businesses since it encrypts their most precious data. Ransom ware assault known as WannaCry, all files on affected computers were encrypted by ransom ware, causing widespread damage. The WannaCry infected over 151 nations and at least 201 thousand machines. Ransom ware like WannaCry was devastatingly effective because of the sophisticated encryption techniques and wide-ranging distribution strategies that define ransom ware. Because of an encryption approach is just one of several aspects that can affect whether or not an attack ransom ware is successful. Researchers in the world of cyber security have been attempting to discover the best way to detect ransom ware using state-of-the-art security protocols. That's a rather hefty sum of money detecting methods and models proposed by various researchers. That's why having a trustworthy infection mechanism for the widespread nature of the WannaCry ransom ware epidemic. Malicious ransom ware like WannaCry uses infection strategies includes a wide range of diseases spread by vectors. Therefore, computing technology becomes an essential part of almost every aspect of human life, from basic survival to advanced scientific research and engineering. Since the formal start of the "Information Age" in the early, societies around the world have rapidly shifted from an industrialized, machine-based economy to one centered on Information Technology. Since, more than 40,001 different viruses have been identified, and this number is rapidly growing. These are only two of the many challenges that are being created by the speed with which science and technology are developing. Malware can be detected using either static or dynamic analysis. Dynamic analysis methods are based on watching the code as it executes in isolation in real time. Static analysis is used on decompiled source code to examine things like access levels, modules, and API calls. In this study, we focus on the static approach case, in which an app's source code is assessed statically. The sources and were used to construct a unique feature vector taken from the Java source code of the program. There are 697 features altogether. We categorized them into one of three major classes: The first is that the on Receive () methods used by the Broadcast Receiver components have been added to the model. Malicious programs are more likely to use overridden versions of the Receive () method than legitimate ones. The likelihood of cyber assaults is rising as the reliance on technology and cloud-based services grows. A collapse in cyber security is considered as a big threat to global economic stability as the COVID-19 pandemic continues to spread. It usually takes roughly two days for cybercriminals to breach a company's internal networks. Advanced persistent threats (APTs) are a distinct and significant security concern because they use many techniques to remain undetected while controlling infected networks. Additional products and research have been conducted along two main paths: static analysis and dynamic analysis, in order to combat the malware. Applying either of these methods to a large number of malware samples takes a lot of time. Malware detection has come a long way, thanks to advances in machine learning and deep learning techniques that can extract unique traits from malicious software. This means that a malware-detecting EDR solution, particularly one that incorporates a lightweight image-based malware detection system, can be more effective.

2. Literature Survey

Malware, short for "malicious software," is software with malicious intentions. [1]There is a vast range of malicious software that can be classified by its behavior and the harm it causes, such as Trojan horses, rootkits, worms, viruses, spyware, backdoors, logic bombs, ransom ware, and adware, and. There are many different motivations for launching an attack on a computer system, such as the damage of computer resources, financial gain, the theft of private and secret data and usage of computing resources, the denial of service to the system, etc. [2].Malware development slowed down in the past because of the developing digital age. However, times have changed, and thousands of new dangerous applications are developed every day. More than that the sophistication of new malware rises in tandem with the amount of harmful programs when compared to regular dangers, targeted, zero-day, stealthy, and persistent ones are much more dangerous. Malware that was freely accessible, ran only once, and was widely disseminated [3].Furthermore, the complex encoding and encryption methods. Malware that may change its form, behavior, and payload. Extremely difficult to detect and analyze malware is being developed the anti-malware tools employ them [4] a technique or techniques based on signs or behaviors. This digitally signed. Anti-malware tools are lightning fast and extremely helpful, but virus obfuscation makes it possible to avoid detection [5]. On the other hand, Behavioral techniques perform better than the obfuscation technique. It is time-consuming to create techniques based on observable behavior, though. Malware, which is described as "a

computer program that compromises a target system by infecting the other programs installed on that system," [6] is a major threat to both individual and commercial security. Malware can invade privacy, stop operations, and cause physical damage to systems and networks. Malicious malware known as ransom ware encrypts data and refuses to unlock the user's computer until a ransom is paid [7]. Malware can be generated and transmitted in a variety of methods, such as by being hidden within legitimate software, pretending to be a safe and useful program, or by phishing assaults. Infection via file less malware can be avoided by adhering to safe browsing practices, such as avoiding questionable links into installing suspicious software or opening suspicious links. It's also crucial to keep your software and operating system up to date and use a reputable antivirus program detection of file less viruses. Antivirus software is a given when talking about safety, and it's assumed that it will [8]don't count on signatures, and they don't employ heuristics. Therefore, the detectability of file less malware is a result of its stealthy nature potential [9]. In addition to a well-configured network and devices, a spot to keep nosy neighbors at bay and keep a look out for anything out of the ordinary. Many diverse fields have seen great success with deep learning and other types of machine learning detection of file less malware [10] is one such field. Recently, machine learning methods have been widely adopted for malware detection [11]. Static Analysis based on signature-based patterns, permissions, and components was previously used for malware detection, [12] however this approach could only detect known malware. Using machine learning methods, even the most cutting-edge and previously undetected malware can be identified. Machine Learning is putting the dataset to use by training and testing models to further research and enhance outcomes. The processes of Machine Learning and Feature Mining will be dissected in their own articles. The vast majority of Big Data consists of unstructured, heterogeneous information stored in a wide variety of file formats (text documents, images, audio/video recordings, etc.) [13]. In contrast to structured data, which has a predetermined exterior shape, unstructured data has its own internal structure. Text files and XML files, each of which have their own unique structure, make up the Android Files. During the phase of Android file collection, 15505 malicious files and 4001 benign files were gathered. The vast number of Android's files makes management a big data problem, and decompiling the source code is a challenging operation. To that end, we'll be decompiling Android code in a manner able to process massive volumes of information. According to the literature [14], many academics have attempted to decompile APK files using a wide variety of methods. The tools' incapacity to extract Java code and XML files in tandem was one barrier; another was the high degree of manual labor and latency inherent in using offline tools. The unstructured data in Android Files once again demonstrated the inadequacy of offline methods. Given these challenges, we need a tool capable of decompiling the file. [15]As a result, we advocate for a simple and automatic recompilation tool that can be accessed by everyone. To accomplish this, we have reviewed the tools and techniques used in the earlier work. The release of source code also inspired the development of other malware families and Internet of Things (IoT) assaults [16]. Protection of insecure Internet of Things (IoT) gadgets and the avoidance of IoT-based attacks are now major challenges. There are reports of experiments with high detection sensitivity in the secondary literature [17]. These were revealed by studying the opcodes and CFGs of the IoT malware, two major parts of the malware's code structure. However, factors such as central processing unit size were not taken into account in the related investigations architecture, and the evaluation's test results class differences, limited population, etc. which can result in biased conclusions. What's more, techniques are viewed as economically inefficient because to acquire opcode sequences, [18] one must resort to high-level inversion. Mechanical devices for design. Mainly focused on identifying and classifying polymeric malware. The most defining features of this type of malware are known to undergo constant evolution. It is because of this that traditional signature-based methods fail to identify the infection. Obtaining the pattern of behavior of different viral strains is the first step in using static or dynamic analysis. The next step was to use many Machine Learning techniques to create a social-based model for detecting and categorizing malware. You will find SVM, KNN, Naïve Bayes, J48 Decision Tree, and MLP among these approaches. Malware is classified using conditional probability in the Naïve Bayes approach. When it comes to Detection Rate, the Multi-Naïve Bayes method with the 'Bytes' attribute is tops, and when it comes to False Positive Rate, the Signature Method with strings is at the bottom. This system's exceptional malware detection and classification skills are the result of the integration of all of these methods. Researchers and security experts have proposed a variety of methods, such as signature-based [19], and machine learning-based, and behavior-based, for detecting and blocking the installation of malware in PE files on Windows systems. Signature-based detection[20] utilizes the unique characteristics of known malware to detect and avert threats. In contrast, behavior-based

detection monitors software for anomalous behavior that may indicate malicious intent. The current file's signature is compared to the signatures in the database of known malicious software to see if any matches are there. How likely a file is to contain malware is based on how similar its signature is to those already collected? Since this technique uses already gathered and validated signatures, it can only detect previously known malware. These strategies used to be deemed reliable, but since malware evolves over time, they are no longer seen as reliable [21]. Because of this, the current study is focusing on methods of malware detection that rely on machine learning. Primarily aims to identify and remove malicious websites and applications. Several signature-matching and machine learning-based identification systems are now in use. [22] The data used to train the Random Forest Model came from several files' Executable headers. Next, a particular file is evaluated using the acquired model to determine its maliciousness. With the correct datasets, we trained a Logistic Regression Model to deal with URLs. In order to get characteristics from the URLs, the regression model has used a tokenizer. [23] The next step is to train the regression model using these values. After training, the model can detect whether a URL is secured or not. After all these components are combined, the final application is created. With this programmer running, the entire system will be protected against malicious applications. [24] The work conducted focuses on the capability of machine learning to detect and classify different types of malware. Information on various forms of malware has been gathered through various means. Image Version, Debug Size, Export Size, Resource Size, and Number of Sections are some of the available options. Malware classification has made use of three ML algorithms. Multiple Support Vector Machines (SVMs), Decision Trees, and Multilayer Perceptrons are the mentioned ways. [25] The employment of multiple binary SVMs in the Multi SVM Algorithm ensures accurate multi-class categorization. It is safe to assume that the model based on neural networks had the greatest performance here. This model gets an accuracy of about 98% on the training dataset and 99% on the test dataset. Another idea states that the most essential quality, which also happens to have the strongest association, is Debug Size.

3. Proposed System

Malicious software, commonly known as malware, is software with malicious intentions, such as stealing information, damaging files, or compromising a system in some other way. The safety of computer networks depends on the accuracy with which harmful software can be detected. Malware is designed to harm a system in various ways, such as by exchanging data with remote hosts, downloading data from other hosts, and then installing that data in the system directory. These behaviors are typical of API calls performed by malicious files after they have loaded DLLs from the system.

3.1 Data Preprocessing

3.1.1 Normalization

By standardizing features, the learning system can avoid performing unnecessary gradient alterations that could slow convergence. With $\rho = 0$ and $\tau = 1$ signifying the mean and standard deviation of a normalized feature vector and its respective values, respectively, the feature vectors are typically normalized separately such that they all follow the same normal (or Gaussian) distribution. A machine learning algorithm's training will be aided by this. This is how we normalized all of the feature vectors in our study before feeding them into the model.

3.1.2 Regularization

Overfitting is a problem that needs to be addressed in machine learning algorithms. It is, once again, of the utmost importance that a trained algorithm produces consistent results when applied to data that wasn't used during training [22]. Dropout and batch normalization are two common approaches. For the sake of this analysis, we choose a dropout rate of 0:16, which means that 16% of the outputs from each hidden layer are not used. Keep in mind that dropout helps only in the training phase and has no use in testing or production [26]. The MLPdf model's learning and generalization abilities are enhanced by batch normalization and training with a batch size of 64. The goal is to rescale the input to each layer so that the mean is zero and the variance is one, similar to how feature vector normalization for the input layer works. To aid in the detection of overfitting and the implementation of model selection during training, a validation dataset consisting of 21% of the training dataset is used.

3.2 Feature Extraction

Static and dynamic analysis are used to extract information from executable files. In this research, they take advantage of the similarities between malicious and clean executables to extract N-gram characteristics and Windows API calls.

3.2.1 N-Gram

A program's n-gram features can be thought of as sequences of n-gram substrings. The n-gram technique's strength lies in its ability to count how often phrases of a given n-gram length appear in text. According to our research, the optimal number of n-grams is 5. This means that each sequence must be exactly 5 bytes in length. They use an n-gram feature extraction technique to get the n-gram features.

3.2.2 Window API Cells

Monitoring API requests can reveal malicious conduct. The API catalog is included in the PE format of executable files. The Portable Executable (PE) header contains information on how a program handles system resource. We disassemble the binary file with the use of Interactive Disassembler Pro (IDA Pro), the gold standard in disassembly software, to inspect and extract the Windows API calls. Using IDA Pro, you may easily disassemble both executable and non-executable files (ELF, EXE, PE, etc.). By automatically recognizing API calls for various compilers and providing hooks to execute user-defined plugins, powerful analysis and control may be applied in a variety of ways [23]. In order to create an IDA database, IDA Pro reads the file into memory and analyzes it. The IDA database files are compiled into a single IDB file (.idb) by IDA Pro once the binary has been decompiled and analyzed. The IDA Pro API allows users to create their own plugins and access the program's internal resources. The.idb database was generated using the idapython system, which immediately runs the disassembly module. The ida2sql plugin is used to transform binary analysis databases from the.idb format into the MySQL database format (.db). The ida2sql plugin generates 16 tables (containing Address reference, Address comments, callgraph, control flow graph, Basic blocks, data, function, etc.) to represent each binary executable. Each representation has its own unique manner of describing the binary data. For instance, the Function table catalogs all the established API system calls, along with the names of any undiscovered functions and their corresponding lengths (the offsets from the start and finish of the function). The instructions table contains all the operation codes (OP), along with their corresponding addresses and block positions.

They gathered the API functions using a table called "Function." Microsoft's Developer Network (MSDN) is used to compare and locate compatible Windows APIs. A developed software compares the API from MSDN with the API calls produced in the database for the malware sample set. The API calls made by malicious code are cataloged using machine opcodes like Jump and Call, together with the nature of the associated function.

3.2.2 Refinement

After n-gram feature extraction, the most relevant features are selected based on the classifier accuracy as a function of the number of features. Then, the best feature selection approach is assessed. Here, we use Principal Components Analysis (PCA) to choose features. One way to speed up calculations is by using principal component analysis (PCA), which can minimize dimensionality. The technique finds the small set of orthogonal linear combinations of the original variables with the maximum variance and uses them to generate a small collection of uncorrelated variables from a vast set of variables[27]. In order to enhance the accuracy of detection and classification, we use the Class-wise document frequency (DCFS) based feature selection method to separately retrieve the API calls for each malware category.

3.3 Training the Model

3.3.1 CNN

For classification purposes, input data are transformed into class scores based on the activations of neurons in a convolutional neural network (CNN), which is made up of neurons structured in a 1-, 2-, or 3-dimensional space (width, height, depth). In contrast to traditional Neural Networks (NNs), just a subset of the neurons in a given layer will be connected to the neurons above it. CNNs often have the following layers in addition to the input and

output layers: FC, RELU, CONV, and POOL. In this piece, they will briefly discuss the structure of the CNN layers that were employed in this proposed approach.

3.3.2 Input Layer

The input image is a 228 x 228 square with a single-color channel. To facilitate future comparisons between this study and others, the image resolution was reduced to this size.

3.3.3 Conv Layer

In this layer, users are provided with the filter numbers, filter widths, and padding necessary to compute the output of neurons corresponding to local regions in the input image. This particular CNN architecture consists of three CONV layers. The first CONV layer consists of 16 2x2-square filters. The amount of padding is the CONV layer and the link it has with the input layer. The larger abstractions that must be tracked necessitate a higher number of filters, or neurons, which grows from 16 to 64 as the network depth increases.

3.3.4 BN Layer

It is positioned between the CONV and ReLU layers, both of which are non-linear, to speed up the learning process. It aids in reducing the noticeable effects of minor data fluctuations.

3.3.5 Relu Layer

This nonlinear activation function is placed right after the batch normalization layer, and it performs a threshold operation on each input element, swapping out any numbers below 0 with 0 and any numbers above 0 with u. Identical to the first assertion.

$$h(u) = \begin{cases} u, & u \geq 0 \\ 0, & u < 0 \end{cases} \quad (1)$$

3.3.6 Classification Layer

The structure of a convolutional neural network ends with this layer. Loss is determined by assigning each input to one of two classes, with the likelihood coming from the Softmax activation function. Formula (2) indicates that the loss can be computed by applying the cross-entropy function to a set of k distinct classes.

$$S(\ominus) = -\sum_{l=1}^{m}\sum_{r=1}^{p} q_{rl} In\, z_r(u_l,\ominus) \quad (2)$$

The result for sample l is represented as $z_r(u_l,\ominus)$, where $\ominus$ is a vector of parameters and q_{rl} is a flag indicating that sample l is part of category r. It is convenient to think of the network's probability that the l th input belongs to class l as $K(q_r = 1\,|u_l)$, where $z_r(u_l,\ominus)$, is the output. In contrast, the RBF-Based SVM classifier is utilized to perform the classification in this investigation, and its implementation details are provided below. However, there is some debate over how many CNN layers should be used. This study scenario actually fits the bill nicely.

Following a brief overview of the CNN, its features will be discussed. Activation for each CNN neuron m_l is calculated as the dot product of its weights v_l with the activation input u_l plus a bias a, as shown in (3).

$$m_l = h\left(\sum_{l=1} v_l, u_l + a\right) \quad (3)$$

Unfortunately, the available training data were too little to enable the adoption of a full-stack CNN for the purpose of distinguishing between benign and malignant breast cancer in mammograms [24][28]. Due to the large amount of data for both classes, the CNN was trained from scratch to recognize the difference between axial and sagittal spine images. The original job, breast cancer categorization, will be pre-trained using the CNN's trained parameters. The newly built spine CNN layers have been trained on relevant picture properties that can be used to finish an image classification procedure. To proceed with a breast cancer classification challenge, we must now incorporate the learned spine CNN layers into a separate CNN. Only the last three layers (FC, Softmax,

Classification) of the spine CNN were modified and cloned into the new breast cancer CNN because of the small size of breast cancer training pictures.

3.3.7 FC Layer

Every single neuron in the layers below it is linked to every single neuron in this FC layer. That is why the FC layer incorporates all the features that the layers below it have obtained. The number of FC layers' outputs must precisely correspond to the number of classes in the target data. Since sagittal and axial spine images require classification, the output size of the FC layer parameter was set to 2.

3.3.8 RBF-SVM Classifier

For its classification prowess in linear and non-linearly separable data, Support Vector Machines (SVMs) have been commonly used in circumstances with a little dataset to train on, like the one used in this work. The RBF-Based SVM was used to compensate for the CNN's deficit in distinguishing whether a case was benign or malignant, as well as to provide more robust findings to make up for the CNN's requirement of a large number of training samples to generate a satisfying result. Data is separated linearly in RBF-Based SVM because the non-linear kernel function maps the input space to a higher-dimensional feature space. Function of the RBF-based SVM is depicted in (4). Input space feature vectors are denoted by u and z. $\|u-z\|^2$The kernel parameter τ^2 is equal to the square of the Euclidean distance between the u and z vectors.

$$p(u,z) = exp\left(-\frac{\|u-z\|^2}{2\tau^2}\right) \qquad (4)$$

There is considerable discretion in fitting the data samples and in setting the decision boundary between benign and malignant circumstances thanks to the width parameter (τ) of the RBF kernel. It's important to note that leave-one-out validation was used on the training data to determine the best-performing parameter τ. Finally, features recovered from the FC layer of the retrained CNN were used as inputs to the RBF-Based SVM classifier, which was then used to label cases as benign or malignant.

4. Result and Discussion

Malicious software has become a serious threat to computers ever since their widespread use of the Internet. By encrypting the file system table of the infected computer's hard disk and stopping Windows from booting, the virus effectively destroys the user's machine. If the victim wants to decrypt and regain access to the system, the attacker will demand a "ransom."

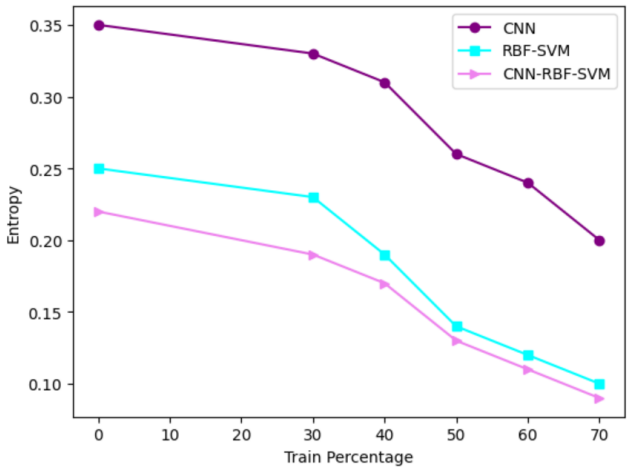

Fig. 1. **Cross-Entropy of Different Detection Modules**

By averaging the entropy of the posteriori probability for each test case, the average cross-entropy of a classifier is displayed in Figure 1. Cross-entropy tends to diminish as training set size increases. For the largest training set, Mal-ID's basic algorithm achieves the lowest a posteriori cross-entropy.

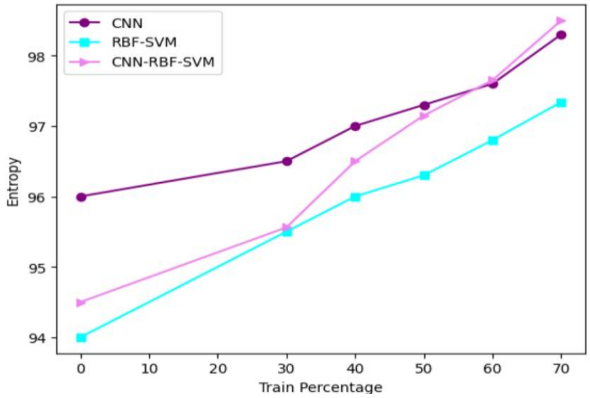

Fig. 2. **Comparing Accuracy Performance of the Model**

In Figure 2, we can see how the RBF-SVM and CNN-RBF-SVM models compare to the accuracy of the CNN basic model. The accuracy grows approximately linearly with the size of the training set, which agrees with our expectations. RBF-SVM performs better than the alternatives while working with a limited training set. On the other hand, the CNN-RBF-SVM basic model performs optimally in the long run with the greatest training set.

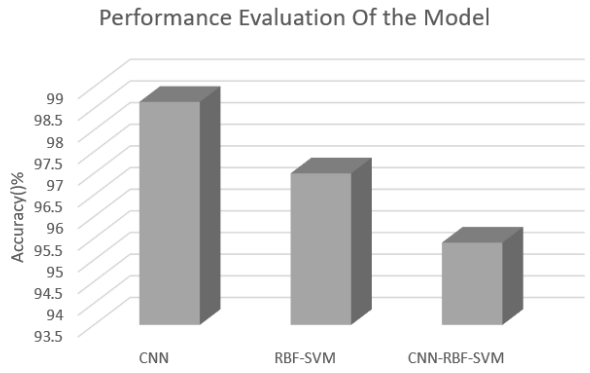

Fig. 3. **Performance Evaluation of the Models**

Figure 3 compares the algorithms' accuracy before and after data normalization. It shows the accuracy comparison of CNN, RBF-SVM and CNN-RBF-SVM model.

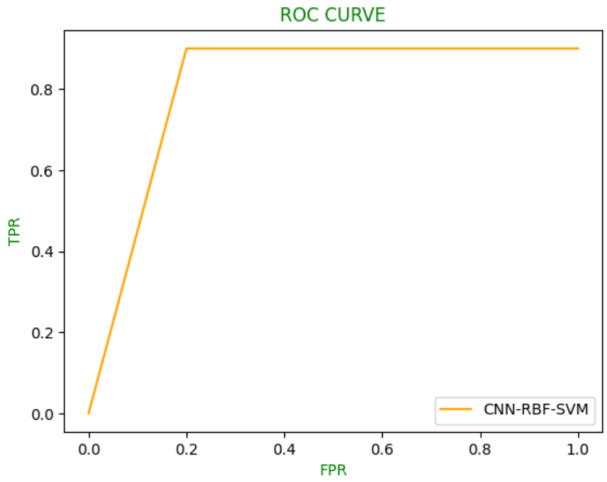

Fig. 4. **ROC Curve of the Model**

The True Positive Rate (TPR) was determined from the initial dataset of 100 samples to generate the ROC curve for the suggested detection method.

5. Conclusion

Smartphone apps providing essential services like banking, healthcare, and mobile commerce have seen an exponential increase in popularity, giving rise to significant safety issues. Application updates downloaded from unofficial sources pose a security risk because they have not been verified. Data leakage, fraudulent deductions for sending premium SMS, acquiring root access to the android system, and other threats are only a few of the many that can be caused by malware-infected apps. Because they rely on signature databases, which necessitate regular updates, current anti-virus systems can't effectively detect zero-day malware. As part of the preprocessing phase, data normalization and regularization are performed. Features must be selected, N-gram features extracted, and Windows API methods used. The CNN-RBF-SVM pedagogical framework is used in this section. The planned strategy is evaluated in comparison to two of its primary rivals, the RBF-SVM and the CNN. The success rate of the proposed method (about 98.5%) is slightly higher than that of the two methods used for comparison.

References

[1] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Comput. Secur.*, vol. 83, pp. 208–233, 2019, doi: 10.1016/j.cose.2019.02.007.

[2] A. Damodaran, F. Di Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 13, no. 1, pp. 1–12, 2017, doi: 10.1007/s11416-015-0261-z.

[3] E. M. Dovom, A. Azmoodeh, A. Dehghantanha, D. E. Newton, R. M. Parizi, and H. Karimipour, "Fuzzy pattern tree for edge malware detection and categorization in IoT," *J. Syst. Archit.*, vol. 97, no. December 2018, pp. 1–7, 2019, doi: 10.1016/j.sysarc.2019.01.017.

[4] and I. S. Alam, Shahid, R. Nigel Horspool, Issa Traore, "A framework for metamorphic malware analysis and real-time detection," *Comput. Secur.*, vol. 48, pp. 212–233, 2015, [Online]. Available: https://eje.bioscientifica.com/view/journals/eje/171/6/727.xml.

[5] W. Zhang, H. Wang, H. He, and P. Liu, "DAMBA: Detecting Android Malware by ORGB Analysis," *IEEE Trans. Reliab.*, vol. 69, no. 1, pp. 55–69, 2020, doi: 10.1109/TR.2019.2924677.

[6] Thangavel, S., & Selvaraj, S. (2023). Machine Learning Model and Cuckoo Search in a modular system to identify Alzheimer's disease from MRI scan images. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 11(5), 1753-1761.

[7] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of Malware Threats and Techniques: A Review," *Int. J. Commun. Networks Inf. Secur.*, vol. 12, no. 3, pp. 326–337, 2020, doi: 10.17762/ijcnis.v12i3.4723.

[8] Kumaresan, T., Saravanakumar, S., & Balamurugan, R. (2019). Visual and textual features based email spam classification using S-Cuckoo search and hybrid kernel support vector machine. Cluster Computing, 22(Suppl 1), 33-46.

[9] A. Afreen, M. Aslam, and S. Ahmed, "Analysis of Fileless Malware and its Evasive Behavior," *1st Annu. Int. Conf. Cyber Warf. Secur. ICCWS 2020 - Proc.*, 2020, doi: 10.1109/ICCWS48432.2020.9292376.

[10] Saravanakumar, S., & Thangaraj, P. (2019). A computer aided diagnosis system for identifying Alzheimer's from MRI scan using improved Adaboost. Journal of medical systems, 43(3), 76.

[11] P. Agrawal and B. Trivedi, "A Survey on Android Malware and their Detection Techniques," *Proc. 2019 3rd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2019*, no. February, 2019, doi: 10.1109/ICECCT.2019.8868951.

[12] P. Agrawal and B. Trivedi, "Analysis of Android Malware Scanning Tools," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 3, pp. 807–810, 2019, doi: 10.26438/ijcse/v7i3.807810.

[13] Saravanakumar, S. (2020). Certain analysis of authentic user behavioral and opinion pattern mining using classification techniques. Solid State Technology, 63(6), 9220-9234.

[14] Saravanakumar, S., & Saravanan, T. (2023). Secure personal authentication in fog devices via multimodal rank-level fusion. Concurrency and Computation: Practice and Experience, 35(10), e7673.

[15] A. Bandi and L. Sherpa, "Android Malware Detection Using Machine Learning Classifiers," *Lect. Notes Data Eng. Commun. Technol.*, vol. 141, no. Ngmast, pp. 191–200, 2023, doi: 10.1007/978-981-19-3035-5_15.

[16] A. Costin and J. Zaddach, "IoT Malware: Comprehensive Survey, Analysis Framework and Case Studies," *BlackHat USA*, pp. 1--7, 2018.

[17] A. Azmoodeh, A. Dehghantanha, and K. K. R. Choo, "Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 88–95, 2019, doi: 10.1109/TSUSC.2018.2809665.

[18] K. K. R. HaddadPajouh, H., Dehghantanha, A., Khayami, R., & Choo, "A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting," *Futur. Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018.

[19] O. Savenko, A. Nicheporuk, I. Hurman, and S. Lysenko, "Dynamic signature-based malware detection technique based on API call tracing," *CEUR Workshop Proc.*, vol. 2393, pp. 633–643, 2019.

[20] A. K. Sahoo, K. S. Sahoo, and M. Tiwary, "Signature based malware detection for unstructured data in Hadoop," *2014 Int. Conf. Adv. Electron. Comput. Commun. ICAECC 2014*, 2015, doi: 10.1109/ICAECC.2014.7002394.

[21] C. R. Panigrahi, M. Tiwari, B. Pati, and R. Prasath, "Malware detection in big data using fast pattern matching: A hadoop based comparison on GPU," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8891, pp. 407–416, 2014, doi: 10.1007/978-3-319-13817-6_39.

[22] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," *CODASPY 2016 - Proc. 6th ACM Conf. Data Appl. Secur. Priv.*, pp. 183–194, 2016, doi: 10.1145/2857705.2857713.

[23] J. Yan, Y. Qi, and Q. Rao, "Detecting Malware with an Ensemble Method Based on Deep Neural Network," *Secur. Commun. Networks*, vol. 2018, 2018, doi: 10.1155/2018/7247095.

[24] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11297 LNCS, pp. 402–411, 2018, doi: 10.1007/978-3-030-04780-1_28.

[25] S. Sharma, C. Rama Krishna, and S. K. Sahay, "Detection of advanced malware by machine learning techniques," *Adv. Intell. Syst. Comput.*, vol. 742, pp. 333–342, 2019, doi: 10.1007/978-981-13-0589-4_31.

[26] J. Zhang, "MLPdf: An Effective Machine Learning Based Approach for PDF Malware Detection," *arXiv Prepr. arXiv1808.06991*, pp. 1–6, Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.06991.

[27] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," *Symmetry (Basel).*, vol. 14, no. 11, 2022, doi: 10.3390/sym14112304.

[28] M. Alkhaleefah and C. C. Wu, "A Hybrid CNN and RBF-Based SVM Approach for Breast Cancer Classification in Mammograms," *Proc. - 2018 IEEE Int. Conf. Syst. Man, Cybern. SMC 2018*, pp. 894–899, 2019, doi: 10.1109/SMC.2018.00159.