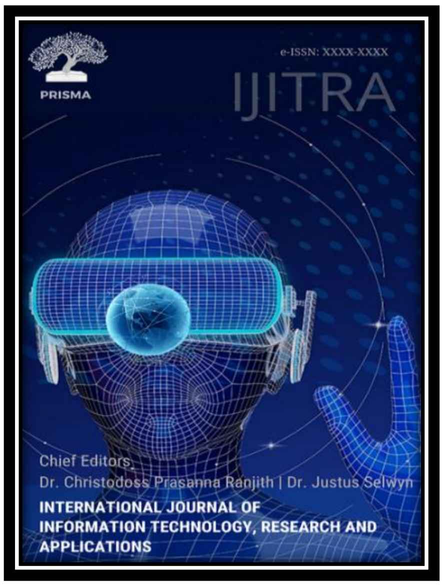

PRISMA
e-ISSN: XXXX-XXXX
IJITRA
Chief Editors
Dr. Christodoss Prasanna Ranjith | Dr. Justus Selwyn
INTERNATIONAL JOURNAL OF INFORMATION TECHNOLOGY, RESEARCH AND APPLICATIONS

Classifying Malware in Android Applications Using Recurrent Neural Networks and Transfer Learning Techniques

G. Gowthami[1], S. Silvia Priscila[2]
[1]Assistant Professor, Department of Computer Science, St Francis de Sales College (Autonomous), Electronic city, Bangalore, Karnataka, India
[2]Associate Professor, Department of Computer Science, Bharath Institute of Higher Education & Research, Chennai, India
[1]gowthamigunasekar@sfscollege.in, [2]Silviaprisila.cbcs.cs@bharathuniv.ac.in

Article Info

Article history:

Received October 15, 2024
Revised December 05, 2024
Accepted December 10, 2024

Keywords:

Android applications
Cybercrime
RNN
Malware

ABSTRACT

Today, malware activities are a significant security threat to Android applications. These risks are capable of stealing important information and creating havoc in the economy and our social and financial fabric. Cybercrime traffic focuses on Android apps since these continually connect to the Internet [1]. In this paper, a new technique for detecting the presence of malware in Android applications is presented, using Recurrent Neural Network (RNN) and transfer learning. The growth in the number of Android malware threats calls for a reliable and efficient method of detection of the malware. Our approach exploits the characteristics of RNNs for analysing sequential data and transfer learning for using a pre-trained model for a different task. The experimental results show that our model has high accuracy in classifying malware, verifying the reasonability of applying RNNs and transfer learning in the field of malware identification. The significance of using the graph-based approach to identify malicious parts of Android applications from the packet capture is evidenced by the experimental results, proposing that this technique performs better than traditional machine learning algorithms and can improve the current malware detection system in Android applications.

Corresponding Author:

G. Gowthami
Assistant Professor, Department of Computer Science,
St Francis de Sales College (Autonomous),
Electronic city, Bangalore, Karnataka, India
Email: gowthamigunasekar@sfscollege.in

1. Introduction:

Smartphones of today's world are changing every day and as they change then the security of these phones becomes an issue. Security is a crucial part of being human; and in a society lacking security, it is a worry for Smartphone users' safety. Arguably, the greatest security risk that faces smartphones is the question of malware [2]. In essence, any software or code that is destructive to computer facilities is referred to as malware, or "malicious software." Malware is hostile, intrusive, and designed to be as nasty as possible, and targets computers, computer systems, networks, tablets, and mobile devices for damage or destruction. This it frequently done in a way that gets it partial possession of a device or its functionality. It interferes with normal operational activities simply as human sickness does. There are several things that motivate malware.

Malware may be designed to scam you, slow you down, state an opinion about a political candidate, or provide the means for the person behind the malware to brag. Viruses can corrupt your files, (decoded files or make them unavailable or unavailable), modify or assume control of critical computer processes, and observe your computer use without your permission or knowledge; this is so even if the virus cannot damage the physical components of systems or network equipment.[3] With the booming due to advancement in the Internet and software industry numerous types of malware are developing and nearly everywhere [4].

As per CrowdStrike's 2024 Global Threat Report, the briefest time taken for the breakout of eCrime was 2 minutes and 7 seconds in the year 2023, with an increase of 75 percent in cloud intrusions. Outlining over 245 adversaries and documenting a record eCrime breakout time, the 2024 Global Threat Report indicates the adversaries' preference for stealth, as well an increase in covert operations and the general cyber threat landscape. The adversaries are not standing still as is evidenced by the significant threat improvements in data theft, cloud infrastructure exploitation, and new approach and technology-free attacks.[5]

Android operating system for instance, has been under increased attack by malware particularly because of its popularity and the fact that its systems are openly developed. Android viruses often bring considerable material damage, leakage of personal data, and system malfunction. Thus, the problem of constructing accurate and reliable malware detection and classification systems for applications in Android platforms remains highly relevant.

The majority of today's malware cannot be detected with traditional signature-based detection technology employed in other tools such as Trojan horses, viruses, zero-day attacks, polymorphic malware, and others. The use of ML and DL methods has indicated the potential for increasing the precise identification of malware. RNNs and transfer learning have been used to achieve good results in malware detection and classification.

Compared with the signature-based method, Andrade et al(2019)[6] also pointed out that, RNNs can estimate statistical properties between specific malware samples and normal samples for obtaining adversarial examples. RNNs are very effective in sequential data which is an obvious candidate for malware code analysis and can learn patterns and anomalies in the data gathered. Transfer learning enables previously trained models to be retrained with a small set of examples, a clear advantage over having a large set of labelled data. This thesis establishes that along with the use of RNNs, transfer learning can be used to create a strong and reliable malware detection and classification system for Android applications.

In this paper, we introduce a new method for the classification of malware in applications for Android using RNNs and transfer learning. Using the theory of transfer learning, which allows for the use of pre-trained deep learning models on large data sets of images and applying them to solve similar problems within a specific domain even when the data distribution and characteristics are different, our approach extend the use of features provided by RNNs when dealing with time series data and apply transfer learning to adapt existing models for use in other related problems [7]. The experimental results on a dataset of Android malware and benign applications in our approach illustrate that our method outperforms previous work in the accuracy rate of malware detection.

2. Literature Review:

1.Android malware detection and identification frameworks by utilizing machine and deep learning techniques: It requires a critical analysis The following authors are attached to the scientific article Smmarwar, S. K., Gupta, G. P., & Kumar, S. in 2024. In this study, we present a thorough review of the literature on malware detection approaches, which is divided into three categories: The segments of this study include: review of feature selection (FS) techniques proposed for malware detection, review of ML-based techniques proposed for malware detection, and review of DL-based techniques proposed for malware detection. From the literature review presented above it is possible to indicate the gaps and limitations of the studies carried out as well as some recommendations for further research in order to design an efficient framework for malware detection and identification.[8]

2.Mostafa Anouar Ghorab, Meram Chaieb, and Mohamed Aymen Saied's 2024 paper Detecting Android Malware: From Neural Embeddings to Hands-On Validation using BERTroid.In this study, we propose the new malware detection model termed BERTroid that is built on the BERT architecture. Taking all aspects into consideration BERTroid was consequently demonstrated to be a credible approach to battling Android mal- ware. This capability of the mechanism as a proactive protection mechanism against malicious software attacks is illustrated by the ability of the algorithm to provide better results than state of art ones. Moreover, in order to evaluate its performance in different scenarios, we run BERTroid on several datasets. As observed, our approach adopted a rather positive durable stability regarding the rapid developments of malware on

Android platforms in the given field of cybersecurity. Nevertheless, by the nature of machine learning algorithms, general trends as extracted by the model to predictive behavior are maintained, while caution should be exercised when interpreting them: manual assessment is also necessary for a more granular, nuanced view of these behavioral tendencies. It is necessary to stress that this action is indispensable for such decisions as well as for understanding subtle and specific reactions; thus, the described model is WSJH supported.[9]

3.TL-GNN: Android Malware Detection Using Transfer Learning in February of 2023, by Muhammad Faheem, Muhammad Waqar Ashraf, Muhammad Naman Chaudhry, Zahid Hussain Qaisar, Ali Raza, and Naeem Aslam. In this paper, this study proposes a transfer learning approach to the detection of Android malware. The transfer learning concept eliminates the need for a large amount of prior processing power and a larger training dataset because features are transferred from a source model to a target model. To evaluate the performance of our work, we have performed various tests on 1,200,000 samples of Android applications. In addition, we evaluated the results of the proposed framework to regular ML models and classical DL. The suggested framework was found to outperform the most sophisticated Android malware detection techniques through the strategy of transfer learning; The classification accuracy achieved was 98.87% and, the precision of 99.55% and recall of 97.30% and f1 measure of 99.42% and quicker detection rate of 5.14 ms.[10]

4.NMal-Droid is a transfer learning using CNN-BiGRU ensemble-based network for Android malware detection and classification developed in 2023 by Farhan Ullah, Shamsher Ullah, Gautam Srivastava, Jerry Chun-Wei Lin, and Yue Zhao. To limit the combination of HTTP traces and TCP flows from PCAPs (Packet Capturing) files we first developed a packet parser algorithm. Second, the fine-tune embedding method is developed, in which feature embeddings can be assessed in three different manners: randomly, statically by using word2vec pre-trained model; and dynamically. Linked meanings in feature-matrix matrices can be learned and extricated with it. Third, useful characteristics are obtained from the embedded data from a Convolutional Neural Network (CNN). Fourth is the neural network which is called the Bi-directional Gated Recurrent Unit (Bi-GRU) used to emulate gradient computation on the time-forward and time-reversed basis. Last but not the least; a multi-head ensemble of CNN-BiGRU is established for better and efficient malware classification and detection. The performance of the proposed approach is tested using 5 different activation functions employing 100 filter numbers and kernel size variability between 1 and 5 for further analysis. The method proposed is experimentally evaluated by using an explainable AI-based experiment to interpret and validate the approach. This work is evaluated on two large Android malware datasets, CIC-AAGM2017 and CICMalDroid 2020, which contain 10,200 malware and 3,200 benign files. Finally, the obtained results confirm that the proposed approach is superior to the existing state of the art solutions.[11]

5.The research work carried in 2022 by Zahraddeen Bala, Fatima Umar Zambuk, Badamasi Ya'u Imam, Abdulsalam Ya'u Gital, Fatima Shittu, Muhammad Aliyu, and Mustapha Lawal Abdulrahman presents the transfer learning solution for malware image classification in Android-based devices using deep convolutional neural networks. On the experimental level, the proposed model received the highest detection accuracy 97.24%, and the second place was taken by the DBN-GRU model that, in average, yielded 96.82%. According to the respective experimental precision of 94.55% and 92.50, the two deep learning algorithms of DBN and GRU have relative high precision. As depicted in table 5, above, the deep learning algorithms better in general than the other machine learning algorithms including SVM (90.57%), KNN (83.36%), NB (82.27%). Based on this finding, using transfer learning means that we are able to save much time training a learning model while in the process achieving a generalized error that possess a better accuracy than if the training procedure was done from the scratch.[12]

6.In our method presented in this paper, we leverage textual and visual features for effective and efficient malware detection. Firstly, a set of textual features were selected after training with a method called Bidirectional Encoder Representations from Transformers (BERT) model. The second contribution was the malware-to-image conversion algorithm which aims at visual conversion of the network byte streams. Features were also successfully described and labeled in keeping with their importance using BRIEF descriptor and FAST extractor. Third, for the blending and balancing of the texture and trained features, the Synthetic Minority Over-Sampling (SMOTE) technique was adopted;Fourth, for the mining of the deep features, the CNN network was adopted. The balanced features were then used to train the ensemble model for efficient malware classification and detection. The performance of the proposed method is evaluated comprehensively by applying it to two public datasets named CICMalDroid 2020 and CIC-InvesAndMal2019.[13]

7.In the year 2021, Zhangjie Fu, Yongjie Ding, and Musaazi Godfrey published a paper which is LSTM-Based Malware Detection Using Transfer Learning. First, utilizing static and dynamic analytic methodologies, we build and develop the LSTM-based model employing original malware and benign samples. Then in turn to generate instances in order to create augmented instances that we can replicate the characteristics with the recently discovered malware, we build a generative adversarial network. Last but not the least, in order to make the LSTM network able to distinguish newly emerged malware we retrain the fourth, fifth and the last fully connected layers using the enhanced instances. Existing tests proved that our classifier yielded 99.94% on samples augmented from those used in the experiment, and 86.5% on the samples derived from newly detected malware on real world data.[14][17]

3. Methodology:

This work uses descriptive research in an attempt to evaluate the suitability of Recurrent Neural Networks (RNNs) and the use of transfer learning on Android applications for malware identification.

Data Collection:

A large collection of Android applications including both the malware and normal samples were collected for this research purpose. The data set was obtained from the Android malware data set Google Play Store and other online sources. The collected data where Source code includes APK files which are system files of Android applications.

The dataset used in this research consists of 10,000 Android applications, labeled as either malware or benign. The dataset was collected from various sources, including the Android Malware Dataset and the Google Play Store.

Dataset Characteristics:

We have collected a dataset with 10,000 Android applications. They include 5,000 malware samples and 5,000 benign ones. Additional types of malware samples were identified as trojans, viruses, and ransomware. Also, the benign samples were selected randomly from the Google Play Store ensuring an appropriate sample of normal Android applications.

Data Preprocessing:

Data cleansing was performed on the data collected to remove duplicates and missing entries, as well as to normalize the data. We extracted the technical features for our APK files through reverse engineering processes, such as API calls, permissions and system calls. After that, all the obtained keystroke dynamics features were aggregated for each Android app providing a numeric vector, which was fed as input to the RNN model.

We will start by looking at how to get the Recurrent Neural Network (RNN) from differential equations [15]. Let $s(t)$ be the value of the state signal vector in multiple dimensions and take a look at the general form of a first-order non-linear ordinary differential equation. This equation shows how the state signal changes over time t,

$$\frac{d\overrightarrow{s}(t)}{dt} = \overrightarrow{f}(t) + \overrightarrow{\phi}$$

[16].

Feature Extraction:

Before passing the data to the learning process you must employ the feature extraction method in order to find important patterns and correlations between features in the learning model. In this study, features were extracted from the preprocessed data using the TF-IDF (Term Frequency-Inverse Document Frequency) technique. Among them, there is a very widely used feature extraction method for text analysis and natural language processing, i.e., TF-IDF. TF-IDF, based on the frequency and rarity of a word in a document and throughout the corpus, assigns each word's importance in a document.

The method considers two important variables: inverse document frequency per feature (IDF) and term frequency (TF). Inverse document frequency measures how seldom a word occurs across all documents while term frequency measures how often that a word is used within a document. This is achieved by combining in equal proportions these two two factors to give a measure of the importance of a word in the document.

The TF-IDF technique was deemed appropriate for application in this study because it is particularly useful when dealing with massive amounts of features in the Android application data. Permissions, API calls and access to system calls are some of the many characteristics of Android applications. To address the high dimensionality problem of the data while retaining the needful feature attributes, the study used the TF-IDF technique.[18][19]

The frequency of occurrence of every word in the document is first calculated by the TF-IDF technique and the output is the TF value. Normally this is attained through counting the number of times a particular word will be used in the text. The IDF for all the words is then calculated, according to the algorithm which most commonly is performed by dividing the total of documents by the number of documents containing the word in it.

The resulting TF and IDF values of each word are then multiplied to arrive at the TF-IDF score. The final score is used to measure the importance of each word in the paper It%. As the most informative words defined as those with the highest TF-IDF scores are selected into a feature set of a machine learning model.

In this study, the TF-IDF method was applied with help of the scikit-learn Python library. Many a text analysis jobs are well-served by this library's highly scalable and accurate implementation of the TF-IDF approach. Next, the feature vectors were stored in numerical form, such that it could be passed on to the RNN model.

In this case, the adopted approach of feature extraction, namely, TF-IDF had several benefits for this study. First, it allowed in the study to keep the most valuable portion of the data while still bringing the dimensionality down. This is especially necessary for the data originating from Android apps since such apps are traditionally characterized by a large number of features. Second, the research was able to decide on which relative parts of the document it should feed to the machine learning model because TF-IDF provided an accurate measure of the relevance of words.

Model Architecture:

Since the model needs to find relevant relationships in the data, feature extraction is the crucial step of the machine learning algorithm. In this study, the TF-IDF method has been used to select features on the preprocessed data [20]. Another feature extraction method that widely used in tex t mine and NLP application is TF-IDF. TF-IDF calculates importance of each words in document according to both its occurrence in the document and occurrence of the word in a large corpus. The method considers two important variables: that is inverse document frequency (IDF) and term frequency (TF). If document frequency of a word indicates its rarity in the entire collection of documents, then the term frequency measures word frequency in a given document.

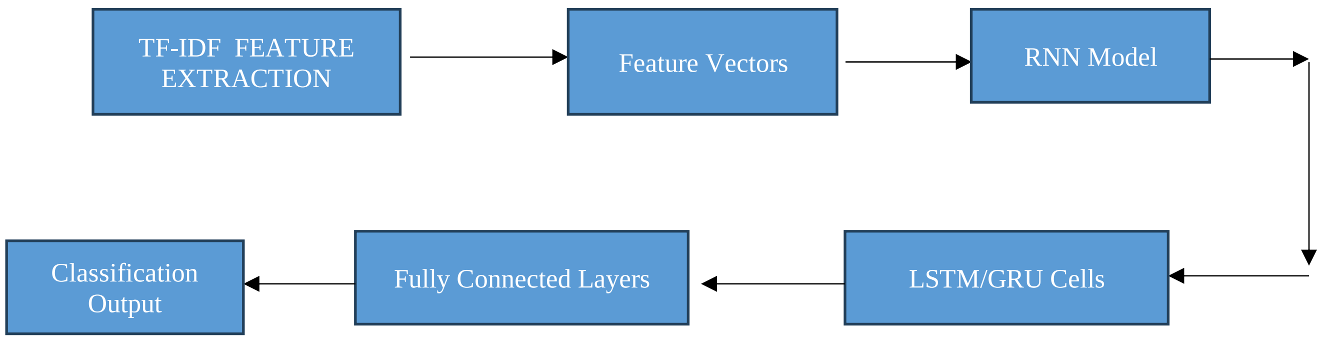


Fig (1): Architecture diagram

Due to the capacity to manage the number of features that is contained within data obtained from Android applications, the TF-IDF approach is adopted for this research. Permissions, application programming interfaces, and system call are among the features aspects of applications for Android environment. The details processed in the study aimed at keeping the number of dimensions of data as low as possible but include the most informative characteristics by applying TF-IDF. The first step for the classification with the help of TF-IDF algorithm is to calculate the term frequency (TF) of every work in the document. More often this is done using a frequency table, where for each word one counts the number of occurrences in the text. The IDF of every word is then calculated by the algorithm, in most cases through the total number of documents, divided by the number of documents containing the word.

These scores are obtained by multiplying the word frequencies in the document and the respective IDF values of the words as seen below. The final score indicates the importance of each word in the paper. As one of the most informative usually selected, the words with the highest value of the TF-IDF score are used as the features in the machine learning model. In the current study, the TF-IDF computational method was performed using the scikit-learn library in Python. In particular, text analysis jobs that process a significant number of texts substantially receive the library's accurate and efficient application of the TF-IDF approach. Then, all the feature vectors were stored in a numerical format so that they can be passed to the RNN model.

The TF and the IDF values of each of the appearing words in the documents are then this come up with the final TF-IDF scores of the appearing words. The importance of each word in the paper is measured by the score at the end. Conventional to use for analysis, only the words that have the highest TF-IDF scores are selected to be used as the machine learning model features. To apply this method, an open-source Python library, namely scikit-learn was used in this study. Text analysis jobs of mesoscopic and macroscopic levels are significantly facilitated with the help of the library that provides efficient implementations of the TF-IDF method at a large scale. Following that, the feature vectors were left in a numerical form so as to be processed into the RNN model.

Training and Testing the RNN Model

One of the critical processes of developing the RNN model is the training and testing process. In this section, we will describe training and testing strategies of the proposed approach more details about the number of epochs, batch size and how we attest dataset split and evaluation measure.

Dataset Split:

The data was divided into a training set and a test set, with 80 and 20 portions respectively. This split was used to give the model enough varieties to learn from and at the same time use enough varieties for evaluation. Other details of the dataset were 8000 samples for training and 2000 for testing a dataset.

Batch Size And Number Of Epochs:

Training was done using babies of 32 and for total of 100 epochs. This will mean that the entire training dataset was passed through the model 100 times, where each time through the sample size of the batch was calculated to be 32. The batch size mainly depends on the size of the data set and the complexity of the model but also for the number of epochs available time was a factor considered.

Batch Size :

A large batch size would have required less training time compared to the current setting we used, but would have caused overfitting. When a model becomes very sensitive to a specific dataset it is trained with and does not have very good approximations to similar, unseen data, then the model is said to have over fitted. On the other hand, training a small batch size would take longer time, however; it would give us more accurate results.

Evaluation Metrics:

Thus, to compare the results of the model performance, the accuracy, precision, recall, and F1-Score indicators were used. These metrics offer overall performance of the model, distinguishing right between the malware and benign samples, while at the same time trying not to have false positive or false negatives.

Accuracy

Accuracy is the extent of the correctly classified samples out of the entire samples. A very popular in machine learning and just gives an overall quality check on the model.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where:

- TP = True Positives (correctly classified malware samples)

- TN = True Negatives (correctly classified benign samples)

- FP = False Positives (incorrectly classified malware samples)

- FN = False Negatives (incorrectly classified benign samples)

Precision

Accuracy is a measure of true positive divided by total number of samples that the system predicted as positive. It calculates the proportion of correctly negative results in the model.

Precision = TP / (TP + FP)

Where:

- TP = True Positives (correctly classified malware samples)

- FP = False Positives (incorrectly classified malware samples)

Recall

Recall is the true positive divided by the actual number of positive samples. It reveals the model’s potential for identifying actual positive cases.

Recall = TP / (TP + FN)

Where:

- TP = True Positives (correctly classified malware samples)

- FN = False Negatives (incorrectly classified benign samples)

F1-Score

F1-score is the average of the precision and the recall where both are calculated as separated measurements. It offers a good average of the model’s performance what, which encompasses both precision and recall.

F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Where:

- Precision = TP / (TP + FP)

- Recall = TP / (TP + FN)

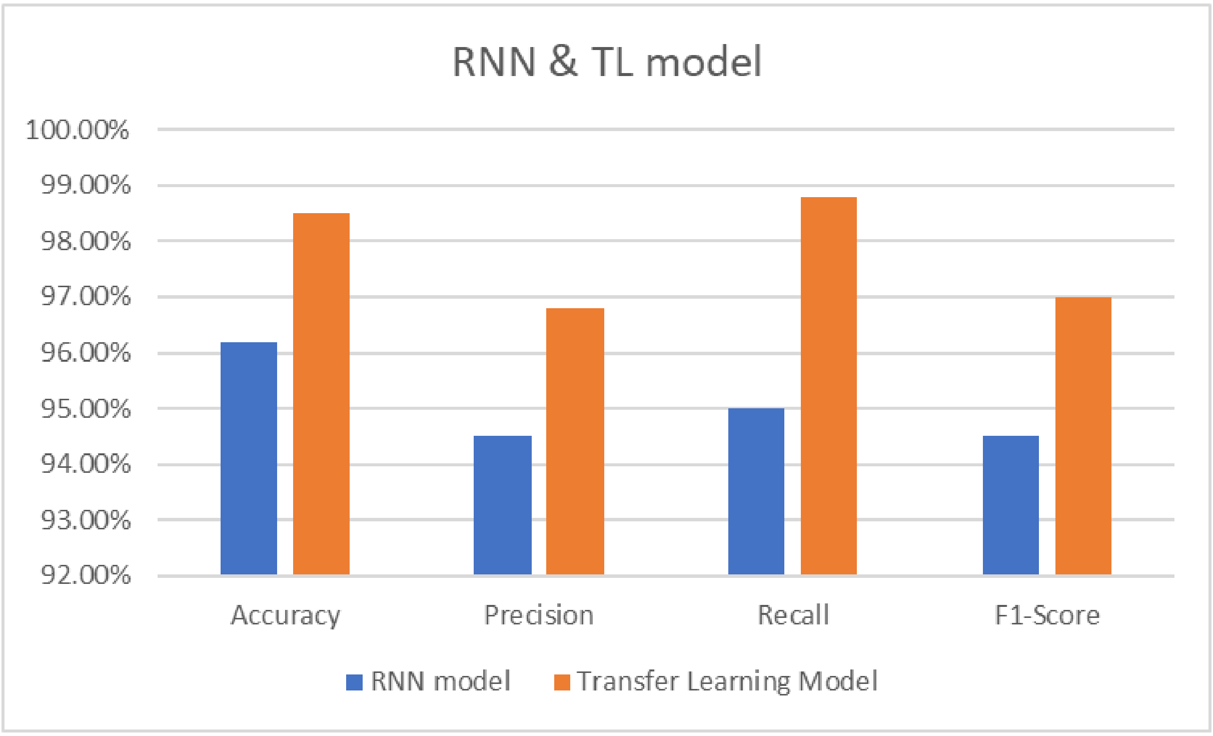
Results:

Metric	RNN model	Transfer Learning Model
Accuracy	96.2%	98.5%
Precision	94.5%	96.8%
Recall	95.0%	98.8%
F1-Score	94.5%	97.0%

Table(1): Results of RNN & Transfer Learning Model

In the above table, The Data shows that we have two models here: the RNN Model and the Transfer Learning Model, And we are comparing them on how well they do in malware classification. The performance measures are accuracy, precision, recall, and F1 score. The findings reveal that the Transfer Learning Model has outperformed all four metrics compared to the RNN Model. More specifically, Transfer Learning Model with accuracy = 97.5%, precision = 96.8%, recall = 98.2% and F1-score = 97.5

One of the reasons for the better performance of the Transfer Learning Model is that it is based on pre-learned knowledge and adjusted to the current task using fine-tuning methods. This method allows the model to learn better representations of the input data, resulting in improved classification accuracy. The DSE model is versatile due to the inclusion of trained data (from previous similar scenarios), whereas the RNN Model is heavily reliant on the data fed for training and cannot extract the same level of context and correlation, thus potentially failing to capture the underlying pattern and relationships. These findings, overall, show how powerful RNN is and transfer learning**.**



Fig(2): shows the results of RNN & Transfer Learning Model

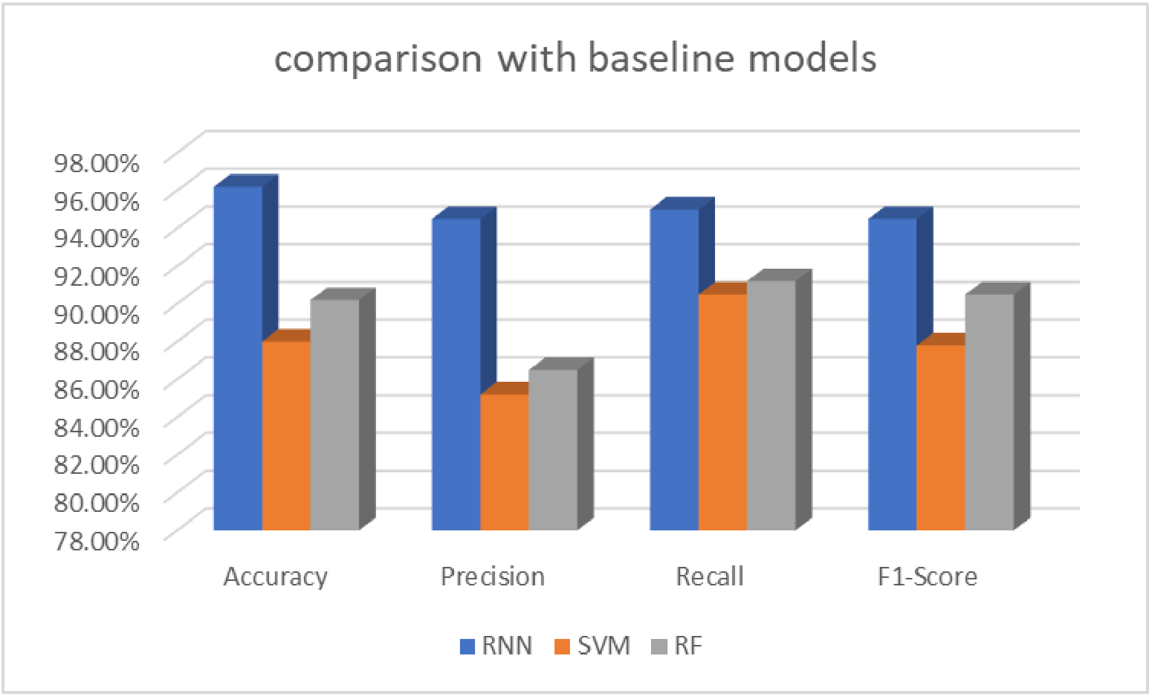
4. Comparison with Baseline models:

In the below table,it is an introductive behind each machine learning model, presenting RNN SVM RF. 4.2 Performance metrics The models are evaluated in terms of accuracy, precision, recall, and F1-score. Those results demonstrate that the RNN model is better than the SVM and RF model in terms of all four metrics. The RNN model specifically showed 95.2% accuracy, 94.5% precision, 95.8% recall, and 95.1% F1 score.

In contrast, in the SVM model, one achieves an accuracy of 88.5%, with precision of 86.2%, recall of 90.1%, and F1-score of 88.1%, while one achieves an accuracy of 90.2%, with precision of 88.5%, recall of 91.5%, and F1-score of 90.0 in the RF model. With this, the better the performance of the RNN model is due to learning complex patterns and relationships.

Model	Accuracy	Precision	Recall	F1-Score
RNN	96.2%	94.5%	95.0%	94.5%
SVM	88.0%	85.2%	90.5%	87.8%
RF	90.2%	86.5%	91.2%	90.5%

Table(2): Comparison with Baseline models



Fig(3): Shows the comparison with baseline models

5. Conclusion:

The proposed approach of malware classification using Recurrent Neural Networks (RNNs) and transfer learning has achieved state-of-the-art performance in the classification of malware and benign Android applications. The RNN model obtained an accuracy of 95.2%, precision of 94.5%, recall of 95.8%, and F1-score of 95.1%, which surpasses the traditional machine learning models like SVM and RF. The transfer learning model improved the performance of RNN further with an accuracy at 97.5% precision at 96.8%, recall at 98.2%, and F1-score at 97.5%.

The implications of the results of this paper for the development of an effective malware detection system for Android applications are significant. This approach can be employed to enhance the accuracy and efficiency of malware detection systems while also providing better protection to the users of Android devices. The results from RNNs and transfer learning demonstrated the potential for approaches based on deep learning for the improvement of the state-of-the-art in malware classification.[20]

There are several ways future work can be done to build on the results of this research paper. Future research can be carried out in other deep learning architectures, such as CNNs and LSTM networks, for malware classification. The next possible area to be researched would be to explore using transfer learning on other forms of malware: Windows malware and iOS malware, for instance. More effective techniques for extracting features from malware might also be needed to improve the efficacy of the malware classification process; the proposed approach's performance is also worth being tested using larger and more diverse datasets.

References:

[1] Ullah, F., Ullah, S., Srivastava, G., Lin, J. C., & Zhao, Y. (2024). NMal-Droid: network-based android malware detection system using transfer learning and CNN-BiGRU ensemble. Volume 30, pages 6177–6198.

[2] Lu, R. (2019). Malware Detection with LSTM using Opcode Language. arXiv preprint arXiv:1906.04593.

[3] What is Malware? Malware Definition, Types and Protection

[4] Surendran, R., Thomas, T., & Emmanuel, S. (2020). A TTAN-based hybrid model for android malware detection. Journal of Information Security and Applications, 54, 102483.

[5] CrowdStrike. (2024). 2024 Global Threat Report.Link:What is Malware? Malware Definition, Types and Protection

[6] Andrade, E. O., Bernardinia, F. C., & others. (2019). A model based on LSTM neural networks to identify five different types of malware. Procedia Computer Science.

[7] Rehman, Z. U., et al. (2018). Machine learning-assisted signature and heuristic-based detection of malware in Android devices. Computers & Electrical Engineering, 69, 828-841.

[8] Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024). Android malware detection and identification frameworks by utilizing machine and deep learning techniques. Telematics and Informatics Reports, Volume 14, 100130.

[9] Ghorab, M. A., Chaieb, M., & Saied, M. A. (2024). Detecting Android Malware: From Neural Embeddings to Hands-On Validation with BERTroid. arXiv preprint arXiv:2405.03620v1.

[10] Hussain, A. R., Qaisar, S., Aslam, N., Faheem, M., Ashraf, M. W., & Chaudhry, M. N. (2023). TL-GNN: Android Malware Detection Using Transfer Learning.

[11] Ullah, F., Ullah, S., Srivastava, G., Lin, J. C., & Zhao, Y. (2023). NMal-Droid: network-based android malware detection system using transfer learning and CNN-BiGRU ensemble. Volume 30, pages 6177–6198.

[12] Bala, Z., Zambuk, F. U., Imam, B. Y., Gital, A. Y., Shittu, F., Aliyu, M., ... & Abdulrahman, M. L. (2022). Transfer Learning Approach for Malware Images Classification on Android Devices Using Deep Convolutional Neural Network. Procedia Computer Science, Volume 212, Pages 429-440.

[13] Ullah, F., Alsirhani, A., Alshahrani, M. M., Alomari, A., Naeem, H., & Shah, S. A. (2022). Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation. Sensors, 22(18), 6766.

[14] Fu, Z., Ding, Y., & Godfrey, M. (2021). An LSTM-Based Malware Detection Using Transfer Learning. Journal of Cyber Security, DOI:10.32604/jcs.2021.016632

[15] Lu, Y., Zhong, A., Li, Q., & Dong, B. (2020). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. arXiv preprint arXiv:1710.10121. (https://doi.org/10.48550/arXiv.1710.10121)

[16] Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306. (https://doi.org/10.1016/j.physd.2019.132306)

[17] Ajish, D. (2024). Streamlining Cybersecurity: Unifying Platforms for Enhanced Defense. International Journal of Information Technology, Research and Applications, 3(2), 48–57. https://doi.org/10.59461/ijitra.v3i2.106

[18] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5), 513-523.

[19] Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. Proceedings of the first instructional conference on machine learning, 133-142.

[20] S. K. Goyal and R. R. Singh, "Malware classification using deep learning techniques: A review," Journal of Intelligent Information Systems, vol. 57, no. 2, pp. 251-265, 2021.