Multi-Model Deep Learning Ensemble Approach for Detection of Malicious Executables

Mohammad Eid Alzahrani
Department of Engineering and Computer Science,
Faculty of Computer Science & Information Technology
AlBaha University, AlBaha, Saudi Arabia
meid@bu.edu.sa

Abstract: Due to the growing significance of the Internet in many facets of our lives, the World Wide Web, which end-users access via web browsers, is evolving into the next platform for those who want to engage in illegal activity for either their own or another person's financial or personal benefit. Among the reported types of attacks, attacks through malicious executables files are still one of the prevalent challenges. Different static and dynamic analysis approaches have been proposed to detect such executables. The challenge with these approaches is that they failed to detect novel attack types in malicious executables. With the dawn of Machine learning, the detection of novel attacks in malicious executables was possible to detect with high accuracy. Deep learning, which is a part of machine learning that works similarly to human neurons, provides a way to achieve much greater accuracy compared to machine learning. In this study, we propose a stacking-based ensemble approach combining CNN, LSTM, and GRU models to detect malicious executables. The experiment results demonstrate that an accuracy of 99.02% was achieved, which is very high compared to individual deep-learning models.

Keywords: Cybersecurity, Malware Detction, Malacious Executables, Deep Learining.

نهج مجموعة التعلم العميق متعدد النماذج للكشف عن العناصر التنفيذية الضارة

الملخص: نظرًا للأهمية المتزايدة للإنترنت في العديد من جوانب حياتنا ، فإن شبكة الويب العالمية ، التي يصل إليها المستخدمون النهائيون عبر متصفحات الويب ، تتطور إلى النظام الأساسي التالي لأولئك الذين يرغبون في الانخراط في نشاط غير قانوني لأيٍّ منهما. أو المنفعة المالية أو الشخصية لشخص آخر. من بين أنواع الهجمات المبلغ عنها ، لا تزال الهجمات من خلال الملفات التنفيذية الضارة أحد التحديات السائدة. تم اقتراح مناهج تحليل ثابتة وديناميكية مختلفة لاكتشاف مثل هذه الملفات التنفيذية. التحدي في هذه الأساليب هو أنها فشلت في اكتشاف أنواع الهجمات الجديدة في الملفات التنفيذية الضارة. مع فجر التعلم الآلي ، أصبح من الممكن اكتشاف هجمات جديدة في ملفات تنفيذية ضارة بدقة عالية. يوفر التعلم العميق ، وهو جزء من التعلم الآلي الذي يعمل بشكل مشابه للخلايا العصبية البشرية ، وسيلة لتحقيق دقة أكبر بكثير مقارنة بالتعلم الآلي. في هذه الدراسة ، نقترح نهجًا جماعيًا قائمًا على التكديس يجمع بين نماذج CNN و LSTM و GRU لاكتشاف الملفات التنفيذية الضارة. أظهرت نتائج التجربة أنه تم تحقيق دقة 99.02٪ ، وهي نسبة عالية جدًا مقارنة بنماذج التعلم العميق الفردية.

1.Introduction

Computer programs known as "malware" have harmful intentions and are designed to steal personal information, impair the user's network, and harm the operating system [1]. In cyber security, malware detection and mitigation are still works in progress [2]. While many researchers have focused on improving the accuracy and time efficiency of malware detection, not much effort has been placed into developing systems that automatically identify the presence of malware at compile time and avoid further execution. Traditional and paid market antivirus programs (AV) typically use a signature-matching approach. The database must keep a local signature on file to match known malware patterns. Malware may be uniquely identified using signatures, which are brief byte sequences (hashes) with low error rates [3]. However, although malware identification is possible at the compile-time, it could lead to false positives if the application is infected by a polymorphic virus or an unknown malicious program [4].

Malicious PE files are a type of executable file that has the ability to harm computer system. Malicious PE files are commonly used for malware distribution, and some malware creators use them in their attacks [5]. Malicious PE files can be created by programmers who want to make an executable file with malicious intent or by hackers who wish to distribute malware through different channels. Malware that is commonly associated with the malicious PE file includes Trojans, spyware, adware, and rootkits. Malicious file types of malware from Windows Software Development Kit (SDK)When using the Microsoft software development kit (SDK), the file types that can be created are executable files (.exe), libraries (.lib), and plugins (.dll) [6].

The two primary methods for detecting malware are static analysis and dynamic analysis [7]. The static analysis uses portable executable (PE) files to extract certain characteristics without running the code. Using static analysis poses no risk to the user's system, making it a secure option. It is less resistant to malicious executables that are compressed and encrypted Dynamic analysis is more vulnerable to malicious executables that are compressed and encrypted, as it generates features while the executable runs within a sandbox or controlled environment. Dynamic analysis shows the actual nature of the code and is better suited for real-time detection. However, analysis takes longer since the execution route is always the same from run to run. The scientific and anti-malware communities have observed that machine learning and deep learning models offer increased resilience against code modifications in systems used for malware detection[8].

The focus of this study is to create a ensemble of machine-learning models that work together to identify malicious executable files. The approach involves utilizing hybrid features derived from both static and dynamic analysis. The CNN, LSTM, and GRU models will be trained separately to recognize the structural characteristics of malware within the dataset. Finally, an ensemble of these models will be employed to detect malicious executable files.

The paper is structured as follows: Section 2 presents the related work, Section 3 explains the proposed approach, Section 4 describes the experimental setup, and Section 5 provides the conclusion.

2.Related Work

Researchers and experts have utilized domain-level expertise to detect malicious PE files. Feature selection is considered one of the crucial steps in malware detection using machine learning to achieve high detection accuracy. In the process of using Machine and deep learning models for malware detection, the selection of features plays a crucial role. The authors of this study propose a data mining approach [9] to analyze static features, namely PE head, string sequence, and byte sequence, in order to identify malware. Another study [10] introduces a taxonomy for malware detection using machine learning algorithms, describing feature categories, feature selection methods, and ensemble algorithms employed in this research. Additionally, a different approach [11] focuses on malware detection based on automated behavior, utilizing the Anubis online dynamic analysis tool to track collected samples. Classification tasks were performed using machine learning classifiers such as Naive Bayes, decision trees, and k-nearest neighbors. Furthermore, authors [12] present a novel paradigm to classify malware variants into distinct families. The author's prioritized methods for feature extraction and selection. The suggested approach will function on packed and obfuscated samples since the features were retrieved from the content and sample structure. Malware behavior characteristics were used to categorize the data, and fusion was carried out using a per-class weighting paradigm.
Authors [13] discovered that opcode frequency is a distinguishing characteristic of PE files. They used Random Forest as a baseline model and compared it with other deep learning models. To improve performance, [9] proposed a hybrid malware detection method that combines features derived from opcode frequency with dynamic features. This approach outperformed using each set of features individually. Additionally, the process of identifying malware can be viewed as an image recognition problem by converting the binary data into grayscale images and employing various vision-based techniques. In a separate study, [14] utilized high-level gist descriptors and kNNs for malware classification, presenting one of the initial attempts to classify malware using image processing without relying on gist descriptors. A study by [15] demonstrations that the deep Learning (DL) technique produced marginally better results. To illustrate the higher performance, they contrasted DL approaches with feature-based traditional methods like kNN [16]. A general CNN architecture for classifying malware was put out by researchers [17, 18] and tested on well-known benchmark datasets to achieve high accuracy. Instead of focusing on detection, these efforts primarily concentrate on family classification.

Numerous studies have been done on the dynamically acquired opcode sequences from PE files. The lengthy execution time of this approach was its main flaw. An RNN-based method that uses run-time API calls as features was proposed by [19]. Using opcode sequences from the assembly code, an opcode vocabulary is first generated using an LSTM-based technique that was described in [20]. Following that, CBOW embeddings [21] were produced using this and passed through a two-stage LSTM model. By using several parallelization strategies, they decreased the prediction time. Accounting for large opcode sequences of varying lengths created, which results in information loss and the ensuing delay in obtaining them and training, is one issue with these systems [22].

The contribution of the study is to propose a stacking-based ensemble approach that combines CNN, and GRU models to detect malicious executables. The researchers address the challenge of detecting novel attack types in malicious executables by leveraging the power of deep learning, specifically deep neural networks.

3.Proposed Ensemble Model

3.1 Dataset

The dataset used in this study was downloaded from Kaggle.com [23], and prepared by UCI. Features from both malicious and non-malicious Windows executable files are included in the dataset. The dataset contains 373 samples which include 301 malicious and 72 non-malicious files. There are 531 features in this dataset.

3.2 Feature Selection

In this study, a dataset consisting of 531 features was utilized, necessitating the need for feature selection to obtain the most optimal feature set. The Chi-Square feature selection method was employed for this purpose. Chi-Square feature selection is a statistical technique used to identify the most relevant features within a given dataset. It is also known as the chi-square test for independence [24]. This method helps determine the subset of features that are most predictive for the target variable while eliminating those that are less predictive or irrelevant. The tool used for performing chi-square feature selection in this study was WEKA, a Java-based machine learning software suite developed at the University of Waikato in New Zealand [25]. WEKA provides a range of tools for pre-processing, including classification, regression, clustering, association rules, and feature selection. Two scenarios were examined during the feature experiments in this study: the first scenario utilized all the available features in the dataset, while the second scenario employed the selected features obtained from the chi-square test.

3.3 Ensemble Approach

The ensemble of deep learning models is a group of neural networks that work together to improve the detection accuracy of malware attacks. This means using multiple models instead of just one. The advantage of this approach is that it can help to improve accuracy by combining the strengths of different models. The models are developed by training them on different datasets and then combining their predictions. In a wide range of tasks such as image classification, object detection, and text classification, ensembles of deep learning models have demonstrated superior performance compared to conventional machine learning approaches. This can be accomplished by training one model using random data, called a "holdout dataset." Then the model is evaluated against the holdout dataset. This process can be repeated with other models to generate a consensus among the different models. The architecture of the proposed ensemble approach is shown in Fig. 1.

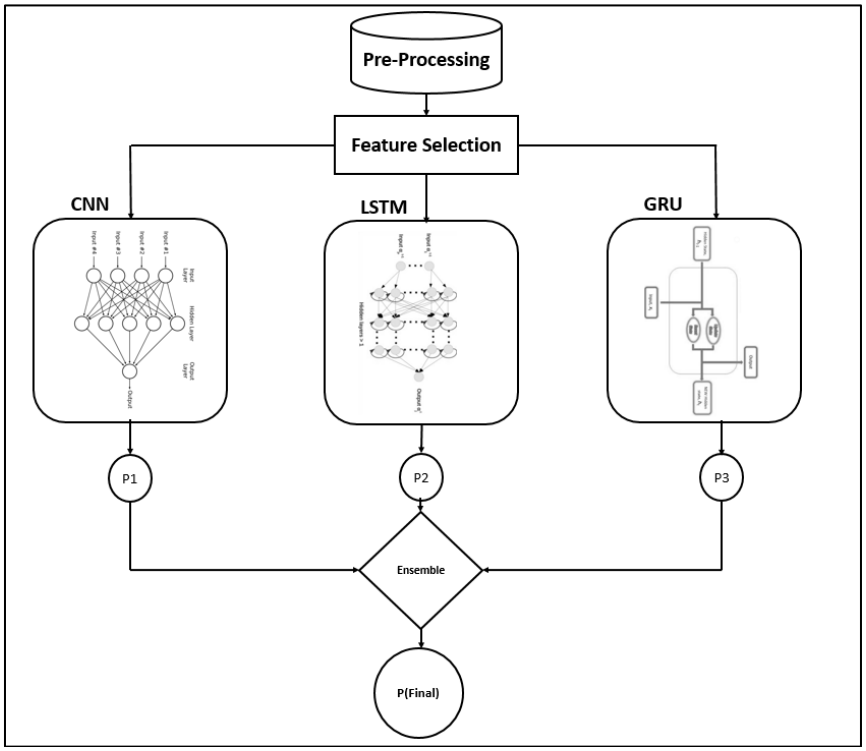

Figure 1: Architecture of the Proposed Approach

3.3.1 CNN

Convolutional neural networks (CNNs) are deep learning architectures primarily designed to recognize patterns. A CNN comprises five layers: the input layer, the convolutional layer, the subsampling layer, the fully connected layer, and finally, the output layer [26], as shown in Fig 2 [27]. The input layer is where all of our data comes.

The convolutional and subsampling layers process this data by performing convolutions and subsamples on it. The fully connected layer then takes all of these processed images and combines them into a single output image using many different combinations of weights. Finally, we have our output layer responsible for taking this final combination of weighted outputs and making predictions about what each one means. Equation 1-3 shows the working of CNN.

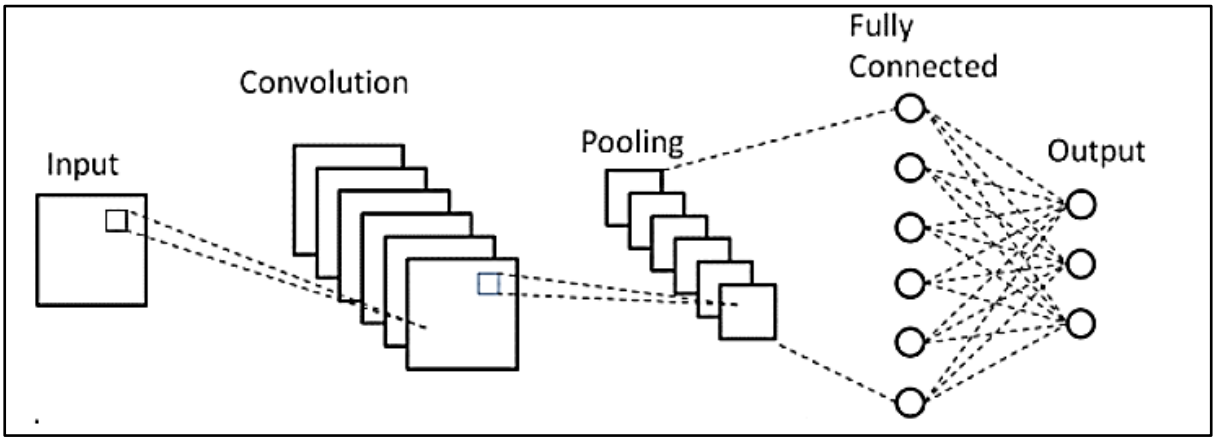

Figure 2: Architecture of CNN

$$x_j = pW_{jxj-1} \quad (1)$$

// x represents the input signal, while as x_j is a subsequent layer, W_j=convolution, p = rectifier or sigmoid

$$x_j(u, k_j) = p\left(\sum_k (x_{j-1}(., k) * W_{j,kj}(., k))(u)\right) \quad (2)$$

$$(f * g)(x) = \sum_{u=-\infty}^{\infty} f(u)g(x-u) \quad (3)$$

3.3.2 LSTM

LSTM networks, a form of recurrent neural networks, possess the ability to acquire knowledge of data sequences and retain them [28]. LSTMs are often used in natural language processing because they can learn to predict words given the context. The LSTM is a type of recurrent neural network or RNN. An RNN is designed to take in a sequence of inputs and produce a sequence of outputs by mathematically modeling the dependencies between inputs and outputs. The LSTM is an enhanced version of the traditional RNN that has improved learning capabilities for long sequences. In order to understand what LSTMs are, it is necessary first to understand the basic difference between RNNs and feed-forward networks. A feed-forward network consists of an input layer, an output layer, and no hidden layers in between.

In a traditional RNN (not the LSTM), there is one hidden layer at the top of a network that receives all inputs as well as one for each output [29]. An LSTM instead has multiple hidden layers at the top— one for each output. There is no input layer, so the only inputs to an LSTM are the previous outputs. The mathematics behind the operation of LSTM is given in Equations 4 to 8. The basic structure of LSTM is shown in Fig. 3 [30].

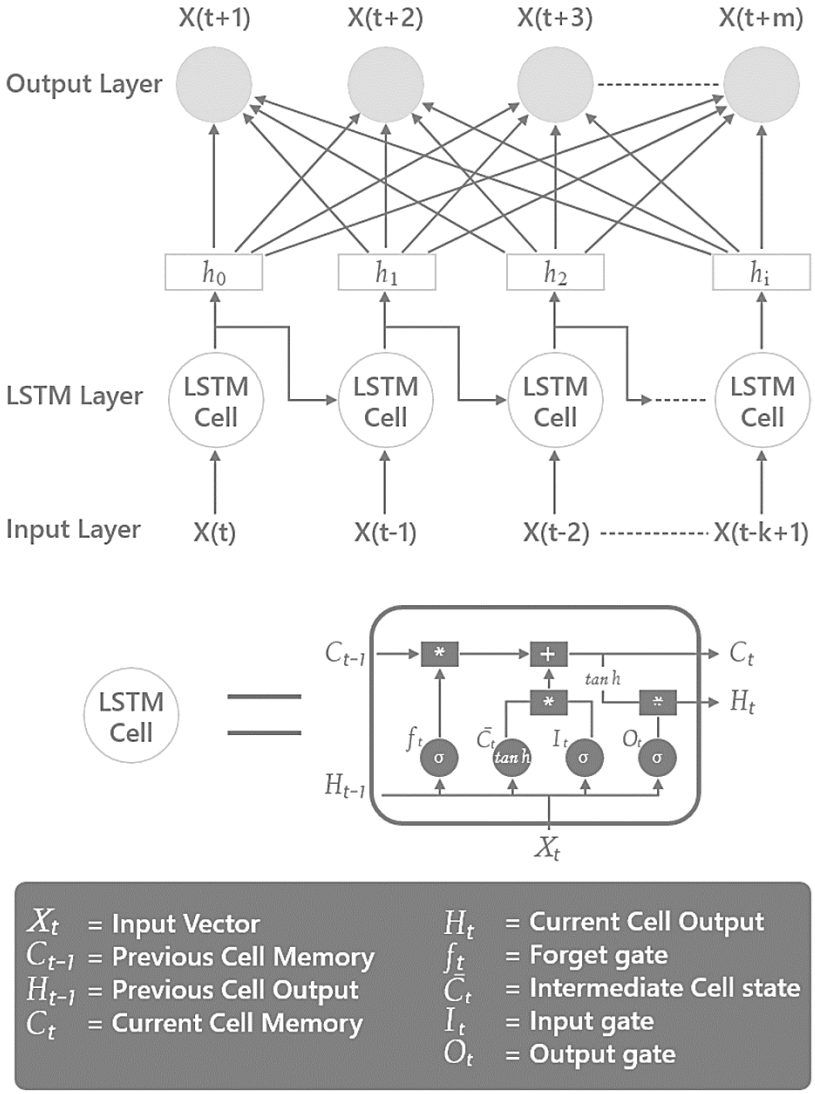

Figure 3: Structure of LSTM

$$h_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (4)$$

$$h_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (5)$$

$$o_t = \sigma(W_0.[h_{t-1}, x_t] + b_0) \quad (6)$$

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c \quad (7)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (8)$$

3.3.3 GRU

The Gated Recurrent Unit (GRU) in deep learning is a type of neural network capable of learning from sequential data [31]. The GRU is more accurate than other types of networks, such as the LSTM [32]. The GRU can learn from sequential data, which makes it more accurate than other types of networks, such as the LSTM. GRU can learn representations that are more abstract than those learned by LSTM networks and is utilized as a basic foundation for more complicated networks. The Gated Recurrent Unit (GRU) in deep learning is a type of neural network that is capable of learning from sequential data, as shown in figure 4 [33]. The GRU is more accurate than other types of networks, such as the LSTM. The GRU can learn from sequential data, which makes it more accurate than other types of networks, such as the LSTM.

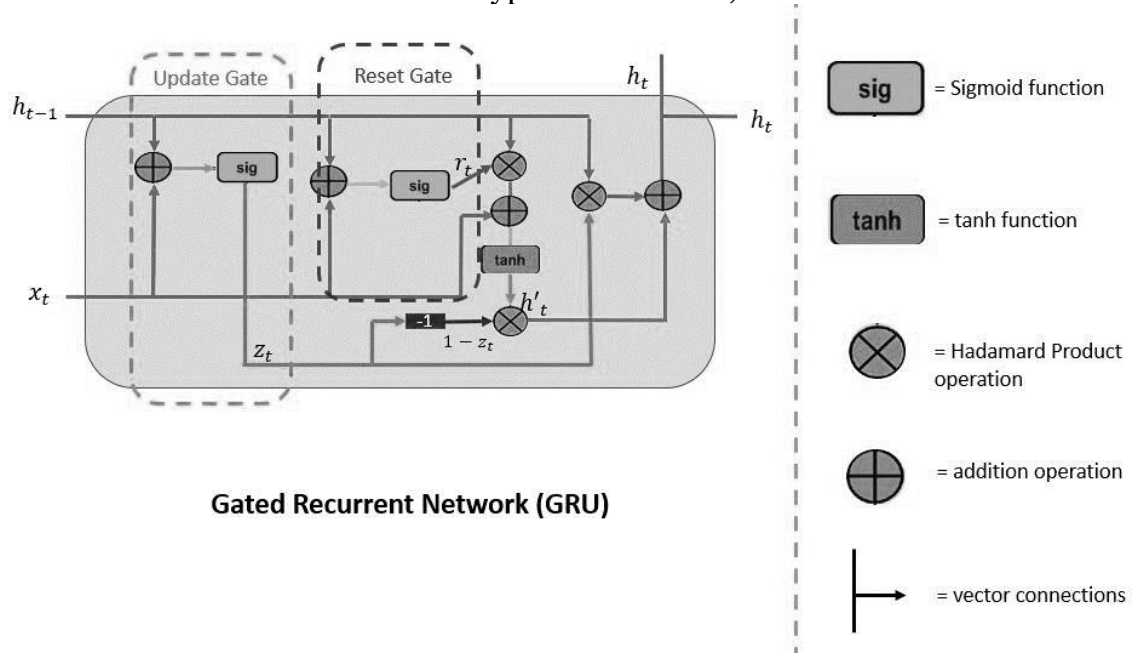

Figure 4: Gate Recurrent Network

Equations 9-12 are used to compute the output from GRU model.

$$z_t = \sigma(W_{xz}x_t + W_{xz}h_{t-1} + b_z) \quad (9)$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r \quad (10)$$

$$\tilde{h}_t = \tanh(W_{xr}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h \quad (11)$$

$$\tilde{h}_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \quad (12)$$

3.4 Stacking Ensemble

Stacking ensemble is a technique used in machine learining and deep learning combine the predictions of different deep neural networks to produce more accurate predictions [34]. The idea behind stacking ensemble is that when using a deep neural network, the error typically increases as the number of layers increases. The idea behind stacking an ensemble is that combining different models can get better results than with one model alone [35]. A deep neural network is a series of layered nodes that

takes an input and then transforms it into an output. The nodes are stacked on top of one another, with each layer transforming the input data into something more complex. A stacking ensemble combines different DL models with different architectures to produce even better results than any single model could on its own [36]. The key advantage of a stacking ensemble is that it may protect a variety of effective models' ability to address classification and regression issues [37].
Stacking, also known as stacked generalization, is an enhanced iteration of the Model averaging ensemble technique. In this approach, sub-models make equal contributions based on their performance weights to construct a new model that generates more precise predictions. This newly formed model, which incorporates the insights from the previous models, is metaphorically referred to as stacking since it builds upon the foundation laid by the older models. To combine the results generated by individual models, a meta-learner is used. The term "meta-learning" in machine learning describes learning algorithms that take inspiration from one another [38]. In the context of ensemble learning, this most frequently refers to the application of machine learning algorithms that discover the most effective way to combine predictions from different machine learning algorithms. In the meta-learning algorithm, at each iteration of its learning process, it utilizes some additional source of knowledge to predict the desired output [39]. This is particularly useful when the task to be learned is very complex and requires significantly more training data than available to achieve good performance. In our case, we used a multi-model such as CNN, LSTM and GRU to create an ensemble based on stacking methodology to get the final classification results.

3.5 Evaluation Matrix

The evaluation and performance evaluation metrics used to evaluate individuals, and the proposed approach are given in Table 1.

Table 1: Evaluation Matrix

Metrics	Formula
Accuracy	Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Precision=$\frac{TP}{TP+FP}$
Recall	Recall=$\frac{TP}{TP+FN}$
F1-Score	F1-Score =$\frac{2\ X\ (Precision*Recall}{Precision+Recall}$

4.Experimental Setup and Results

The experiments for this study were conducted on a PC with an i7, 3.5 GHz processor, 16 GB of RAM, and NVIDIA's RTX 3090 graphics processor. In the first step, the data set was checked for correctness. Then the dataset was shuffled to increase accuracy, and the data was split into 80–20% subsets for training and testing. Then chi-square feature selection was used to get the optimal feature set. Only 165 features out of 531 are used in this experiment when the Chi-square value falls in this given error region using the threshold (alpha from 0 to 0.05). The three models, CNN, LSTM, and GRU,

were first used individually to generate the results based on evaluation matrices, then an ensemble of these models was tested. The experiment was also performed without using the optimal feature set from the feature selection method. The experimental results are depicted in Table 2 without using feature selection, and the result obtained when applying feature selection is in Table 3.

Table 2: Results Obtained without using Feature Selection Method

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CNN	96.32	94.46	95.88	0.96
LSTM	97.61	98.25	97.32	0.97
GRU	96.53	98.84	97.93	0.98
Ensemble	98.84	98.32	98.64	0.99

Table 3: Results Obtained using Feature Selection Method

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
CNN	97.94	95.62	96.48	0.97
LSTM	98.12	99.51	98.52	0.98
GRU	97.39	98.99	98.56	0.97
Ensemble	99.02	99.32	99.68	0.99

The experimental results show that in both cases when the ensemble approached performed better and achieved 98.84% and 99.02% accuracy, respectively, in detecting malicious executables. The selection of activation function and other parameters plays an important role. In this study, the activation function used for CNN was SIGMOID, while for LSTM and GRU tanh activation function was used. The activation function is a node positioned either in the middle or at the end of neural networks. They influence whether or not the neuron fires. Activation functions are the thresholds that change the network's output and are a set of rules that can be applied to a node within the neural network and alter its values. The batch size in training must be greater than or equal to one and smaller than or equal to the total number of samples in the training dataset. It is crucial to select an integer value for the number of epochs that falls within the range of one to infinity. If a neural network model is trained with excessive epochs, it tends to learn specific patterns unique to the training data, resulting in poor performance when applied to new datasets. While such a model may perform well on the training set (sample data), it fails to generalize effectively and performs poorly on the test set. This phenomenon is known as overfitting, where the model loses its ability to generalize beyond the training data. The number of complete passes over the training dataset is referred to as the number of epochs.

In this particular study, the epoch sizes were set at 100 for CNN, 50 for LSTM, and 120 for GRU. Compared to the studies [10-12] our proposed approach perfomed better. This study offers advantages such as high detection accuracy and the ability to detect novel attacks.

5.Conclusion

It is well-recognized that different machine learning and deep learning models have different prediction performances. An ensemble learning approach outperforms a single base classifier by combining numerous separate learning algorithms. As a result, it has become a widely used and successful approach. There are two main problems to be resolved for the ensemble learning methodologies. How to combine is the first problem. base-classifiers that are "fair and distinct." The second is to provide each base -classifier taken into account by the ensemble learning appropriate parameters. We suggest a stack-based ensemble method to detect malicious executables.

The proposed ensemble approach outperformed the individual deep-learning models and achieved an accuracy of 99.02%. Future work of this study include exploring real-time detection, expanding the dataset, addressing evolving attack types, benchmarking against other models, generalizing to other domains, and improving interpretability and explainability of the ensemble approach.

6.References

[1] Patil, B. P., Kharade, K. G., & Kamat, R. K. (2020). Investigation on data security threats & solutions. *International Journal of Innovative Science and Research Technology*, *5*(1), 79-83.

[2] Kapoor, A., Gupta, A., Gupta, R., Tanwar, S., Sharma, G., & Davidson, I. E. (2021). Ransomware detection, avoidance, and mitigation scheme: a review and future directions. *Sustainability*, *14*(1), 8.

[3] Khan, N., Abdullah, J., & Khan, A. S. (2017). Defending malicious script attacks using machine learning classifiers. Wireless Communications and Mobile Computing, 2017.Thambi-Rajah, T., & Jahankhani, H. (2021). The Role of Deep Neural Network in the Detection of Malware and APTs. In *Challenges in the IoT and Smart Environments* (pp. 161-188). Springer, Cham.

[4] Khan, N., Johari, A., & Adnan, S. (2017). A Taxonomy Study of XSS Vulnerabilities. Asian J. Inf. Technol, 16, 169-177.

[5] Case, A., Jalalzai, M. M., Firoz-Ul-Amin, M., Maggio, R. D., Ali-Gombe, A., Sun, M., & Richard III, G. G. (2019). HookTracer: A system for automated and accessible API hooks analysis. Digital Investigation, 29, S104-S112.

[6] Khan, N., Abdullah, J., & Khan, A. S. (2015, August). Towards vulnerability prevention model for web browser using interceptor approach. In 2015 9th International Conference on IT in Asia (CITA) (pp. 1-5). IEEE.

[7] Rathore, H., Sahay, S. K., Nikam, P., & Sewak, M. (2021). Robust android malware detection system against adversarial attacks using q-learning. *Information Systems Frontiers*, *23*(4), 867-882.

[8] Schultz M, Eskin E, Zadok F, Stolfo S. Data mining methods for detection of new malicious executables. In: Proceedings of the IEEE computer society symposium on research in security and privacy; 2001, pp. 38–49.

[9] Shabtai A, Moskovitch R, Elovici Y, Glezer C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Inf Secur Tech Rep. 2009;14(1):16–29.

[10] Firdausi I, lim C, Erwin A, Nugroho AS. Analysis of machine learning techniques used in behavior-based malware detection. In: Second international conference on advances in computing, control, and telecommunication technologies, Jakarta; 2010, pp. 201–203.

[11] Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., & Giacinto, G. (2016, March). Novel feature extraction, selection and fusion for effective malware family classification. In Proceedings of the sixth ACM conference on data and application security and privacy (pp. 183-194).

[12] Rathore, H., Agarwal, S., Sahay, S. K., & Sewak, M. (2018, December). Malware detection using machine learning and deep learning. In International Conference on Big Data Analytics (pp. 402-411). Springer, Cham.

[13] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011, July). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security* (pp. 1-7).

[14] Yajamanam, S., Selvin, V. R. S., Di Troia, F., & Stamp, M. (2018, January). Deep Learning versus Gist Descriptors for Image-based Malware Classification. In *Icissp* (pp. 553-561).

[15] Bhodia, N., Prajapati, P., Di Troia, F., & Stamp, M. (2019). Transfer learning for image-based malware classification. *arXiv preprint arXiv:1903.11551*.

[16] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)* (pp. 1-5). IEEE.

[17] Choi, S., Jang, S., Kim, Y., & Kim, J. (2017, October). Malware detection using malware image and deep learning. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 1193-1195). IEEE.

[18] Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015, April). Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1916-1920). IEEE.

[19] Lu, R. (2019). Malware detection with lstm using opcode language. *arXiv preprint arXiv:1906.04593*.

[20] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[21] Yan, J., Qi, Y., & Rao, Q. (2018). Detecting malware with an ensemble method based on deep neural network. *Security and Communication Networks*, *2018*.

[22] https://www.kaggle.com/datasets/piyushrumao/malware-executable-detection

[23] Sharpe, D. (2015). Chi-square test is statistically significant: Now what?. *Practical Assessment, Research, and Evaluation*, *20*(1), 8.

[24] Weka 3: Machine Learning Software in Java: https://www.cs.waikato.ac.nz/ml/weka/

[25] Abiyev, R. H., & Ma'aitaH, M. K. S. (2018). Deep convolutional neural networks for chest diseases detection. Journal of healthcare engineering, 2018.
[26] MK Gurucharan, Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network Available at: https://www.upgrad.com/blog/basic-cnn-architecture/
[27] Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.
[28] Fan, B., Wang, L., Soong, F. K., & Xie, L. (2015, April). Photo-real talking head with deep bidirectional LSTM. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4884-4888). IEEE.
[29] Chandra, R., Jain, A., & Singh Chauhan, D. (2022). Deep learning via LSTM models for COVID-19 infection forecasting in India. *PloS one*, *17*(1), e0262708.
[30] Patil, S. A., Raj, L. A., & Singh, B. K. (2021). Prediction of IoT traffic using the gated recurrent unit neural network-(GRU-NN-) based predictive model. *Security and Communication Networks*, *2021*.
[31] Hamayel, M. J., & Owda, A. Y. (2021). A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms. *AI*, *2*(4), 477-496.
[32] Gaurav Singhal, LSTM versus GRU Units in RNN, Available at: https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn
[33] Ganaie, M. A., & Hu, M. (2021). Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395*.
[34] Wang, Y., Pan, Z., Yuan, X., Yang, C., & Gui, W. (2020). A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. *ISA transactions*, *96*, 457-467.
[35] Ko, J., Baldassano, S. N., Loh, P. L., Kording, K., Litt, B., & Issadore, D. (2018). Machine learning to detect signatures of disease in liquid biopsies–a user's guide. *Lab on a Chip*, *18*(3), 395-405.
[36] Sesmero, M. P., Ledezma, A. I., & Sanchis, A. (2015). Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, *5*(1), 21-34.
[37] Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. G. (2000, June). Meta-Learning by Landmarking Various Learning Algorithms. In *ICML* (pp. 743-750).
[38] Yao, H., Liu, Y., Wei, Y., Tang, X., & Li, Z. (2019, May). Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference* (pp. 2181-2191).