Malware Detection Using Convolutional Neural Network, A Deep Learning Framework: Comparative Analysis

Nana Kwame Gyamfi[1*], Nikolaj Goranin[2], Dainius Ceponis[3] and Habil Antanas Cenys[4]

[1*]Department of Information Systems, Vilnius Gedimino Technical University, Vilnius, Lithuania.
nana – Kwame.gyamfi@vilniustech.lt

[2]Department of Information Systems, Vilnius Gedimino Technical University, Vilnius, Lithuania.
nikolaj.goranin@vilniustech.lt

[3]Department of Information Systems, Vilnius Gedimino Technical University, Vilnius, Lithuania.
dainius.ceponis@vilniustech.lt

[4]Department of Information Systems, Vilnius Gedimino Technical University, Vilnius, Lithuania.
antanas.cenys@vilniustech.lt

Received: August 02, 2022; Accepted: September 30, 2022; Published: November 30, 2022

Abstract

Malware detection is a quintessential task for every security for securing work stations, mobile devices, servers etc. This detection is mainly used for identifying malware that are causing malicious problems. The traditional detection system has a much lesser rate of detection rate and the chances of getting an error is higher as well. As the emerging technology revolutionized day by day, the usage of Deep Learning (DL) is highly influenced in these detection fields. So, this paper brings an effective DL based detection of malware in which the following are the stages: a) Data collection being carried from Malimg dataset, b) Pre-processing carried out to eliminate the unwanted noise from the dataset and passed to c) Feature extraction, where Principal Component Analysis (PCA) used for extracting required features, d) Feature selection where Particle Swarm Optimization (PSO) used for dimensionality reduction and finally passed for e) Classification where Convolutional Neural Network (CNN) used as a classifier for effective classification. These models are evaluated under measures like Accuracy, sensitivity, specificity, precision, recall, f1-score, TPR, FPR and detection rate over models like VGG16, VGG19, Densenet, Alexnent, Ensemble learning. The proposed system (D-WARE) gives much higher performance with a 96% accuracy.

Keywords: Convolutional Neural Network, Deep Learning, Malware, Particle Swarm Optimization, Principal Component Analysis.

1 Introduction

At the point when the Morris worm at first showed up as a PC infection in 1988–89, antivirus programming applications were intended to distinguish the presence of such malware by coordinating it against an infection definition data set that was refreshed consistently. This is known as signature-based malware location, and it can likewise do a heuristic hunt to decide pernicious action. In any case, new

Journal of Interner Services and Information Security (JISIS), volume: 12, number: 4 (November), pp. 102-115
DOI: *10.58346/JISIS.2022.I4.007*

*Corresponding author: Department of Information Systems, Vilnius Gedimino Technical University, Vilnius, Lithuania.

malware variations exploit antivirus avoidance procedures, for example, code jumbling, making mark-based methodologies incapable of identifying zero-day malware [2]. To pick apart malware using static and dynamic investigation and relegate a signature, a mark-based malware discovery framework requires considerable space level information. Moreover, a mark-based framework takes more time to pick apart malware, which permits an assailant to gain admittance to the framework in that period. Besides, signature-based malware recognition neglects to recognize new types of malware.

Polymorphism and transformation are the most widely recognized muddling procedures utilized by programmers to stay away from signature-based identification, as per security trained professionals. Programming instruments are utilized to physically unload the projects and examine the Application Programming Interface (API) brings to take care of this issue. [3] introduced a computerized framework to separate API calls and break down the unsafe properties utilizing a four-venture method since this technique is asset costly. The malware is unloaded in the primary stage. The paired executable is dismantled in sync 2. Extraction of API calls is the third step. The fourth step is planning API calls and measurable component investigation. This was worked on in [4], which utilized a 5-venture measure that incorporated a profound learning calculation (DLA) like CNN with n-gram highlights accumulated from tremendous examples of both harmless and noxious executables, just as approvals.

The rapid growth of technology has altered daily activities in enterprises and personal lives in this digital environment of Industry 4.0. The development of the advanced idea of the data society has been supported by the Internet of Things (IoT) and its applications. Nonetheless, digital hoodlums assault individual PCs and organizations to take individual information for monetary profit and cause forswearing of administration to frameworks, accomplishing the advantages of this modern upset a major issue. Such aggressors use pernicious programming or malware to represent a significant risk to frameworks and open them to weaknesses [1]. Malware is a PC program that is intended to hurt the working framework (OS). In light of its point and conduct, malware is given various names, for example, adware, spyware, infection, worm, trojan, rootkit, secondary passage, ransomware, and Order and Control (O&C) bot. Malware discovery and relief is a developing worry in the domain of digital protection. Malware creators upgrade their capacity to evade identification when scientists find new methodologies [5-10].

The vital issue with customary AI-based malware discovery frameworks is that they depend on methods like element designing, including learning, and component portrayal, which require a great deal of area information [11][12][13]. Besides, when an aggressor learns the elements, the malware recognition can be crushed [14]. Profound realizing, which is a superior model of neural organizations, has as of late beat conventional MLAs in an assortment of undertakings in the fields of normal language handling (NLP), PC vision, discourse preparing, and numerous others [15]. It looks to catch more elevated level depictions of attributes in profound covered layers during the preparation cycle, with the possibility to gain from disappointments. Profound learning catches new examples and builds up a relationship with recently caught examples to further develop task execution [35, 36]. MLAs experience lessening yields as they see an ever-increasing amount of information, while profound learning catches new examples and builds up a relationship with recently caught examples to further develop task execution.

According to statistics of Cognyt [16], the number of ransomware attacks nearly doubled in the first half of 2021. According to the research, 1,097 organizations were hit by ransomware attacks in the first half of 2021. In contrast, our 2020 report found 1,112 ransomware attacks for the entire year. These attacks involved data exfiltration and the leakage of the victim’s data. Figure 1 shows the most attacks that happened in various industries.

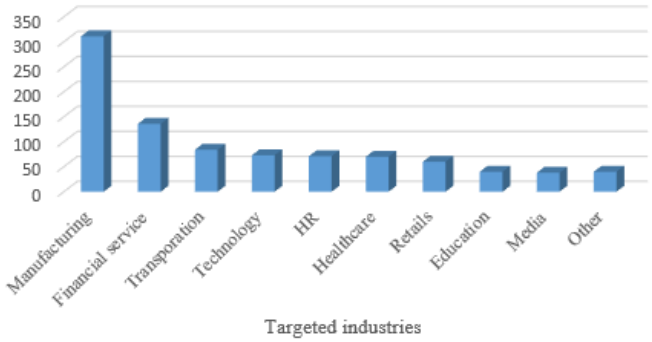

Figure 1: No. of Attacks vs Industries

1.1 Key Highlights

The paper focused on effective deep learning-based malware detection in which following are the objectives:

a. A deep learning-based malware detection (D-WARE) is proposed in which CNN is used as the classifier.

b. PCA and PSO are effectively used as feature extraction and feature selection respectively.

c. Evaluation of this model is depicted using various performance measures such as accuracy, sensitivity, specificity, precision, recall, f1-score, TPR, FPR and detection rate.

d. Also proposed model (D-WARE) is evaluated with other state-of-art models such as VGG16, VGG19, Densenet, Alexnet and Ensemble learning.

Organization of paper: As we previously got over the basic part in Section 1, the rest of the paper is as per the following divisions: Section 2 portrays the writing audit, Section 3 illustrates the philosophy of the proposed framework, Section 4 explains execution investigation and finally concludes with references in Section 5.

2 Literature Review

Shukla et al. (2019) [17] proposed a two-dimensional methodology that can distinguish both customary and covert malware adequately. To start, they removed microarchitectural follows acquired while running the application, which is then taken care of into average AI classifiers to distinguish malware. In equal, they started a computerized restricted component extraction method for compelling subtle malware location, which was utilized as a contribution to Repetitive Neural Organizations (RNNs) for grouping. They put the proposed component under a magnifying glass utilizing covert malware produced with code migration jumbling. The suggested two-dimensional method accomplishes a 94 % exactness, a 93 % accuracy, a 96 % review score, and a 94 % F-1 score. Moreover, when contrasted with CNN-based grouping characterization and secret Markov model (HMM)- based methodologies in identifying covert malware, the recommended procedure accomplishes up to 11% more noteworthy recognition exactness and accuracy, just as a 24 % higher normal review and F-1 score.

Marin et al. (2019) [18] Utilized various portrayals for the information, research the force of profound learning models on the particular test of malware network traffic recognition and arrangement. They

considered crude estimations directly from the surge of observed bytes as the contribution to the proposed models, and assess a few crude traffic highlight portrayals, including parcel and stream level ones, as a significant advantage over the cutting edge. Their discoveries suggest that profound learning models, instead of customary, shallow-like models, can more readily catch the basic insights of malevolent traffic, in any event, while working in obscurity, that is, with no master carefully assembled inputs.

Darem et al. (2021) [19] attempted to accommodate novel malware variants, an Adaptive social-based Incremental Batch Learning Malware Variants Detection model utilizing idea float recognition and successive profound learning (AIBL-MVD) was introduced. The dynamic examination was utilized to extricate malware practices by running malware records in a sandbox and gathering their Application Programming Interface (API) as follows. The malware tests were gathered depending on the malware's first appearance to catch the malware variations' evolving attributes. The basic classifier was then prepared, utilizing a successive profound learning model on a subset of verifiable malware tests. To defeat the disastrous neglecting challenge of steady learning, the new malware tests were mixed with a piece of old information and progressively added to the learning model in a flexible bunch size gradual learning strategy. To identify idea float as a sign for gradually refreshing the model and limiting the recurrence of model updates, the factual cycle control procedure was applied.

Poonguzhali et al. (2019) [20] suggests an approach for detecting malware variants that combines deep learning and a Convolutional Neural Network. Deep learning is a critical component of predictive analysis in today's age. They've turned the harmful coding into grayscale graphics here. The Convolutional Neural Network is used to identify and extract features, and the Support Vector Machine classifier is used to classify the impacted malware images. The malware family to which the impacted code belongs is also mentioned by the classifier. Additionally, they used a bio-inspired optimization technique to deal with the data's imbalance.

A two-layer method is implemented by Feng et al. (2020) [21] to detect malware in Android APPs. The principal layer is a static malware location model dependent on consent, expectation, and part data. It consolidates static provisions with a completely associated neural organization to recognize malware and explore different avenues regarding its adequacy. The principal layer's identification rate is 95.22 %. The result (gainful APPs from the main layer) is then taken care of into the subsequent layer. Another strategy CACNN, which falls CNN and AutoEncoder, is used in the subsequent layer to distinguish malware using organization traffic parts of APPs. In parallel characterization, the subsequent layer has a discovery pace of 99.3 % (2-classifier). Likewise, the new two-layer approach can recognize malware dependent on its classification (4-classifier) and vindictive family (40-classifier). The identification rates are separately 98.2 % and 71.48 % resp. Our two-layer procedure accomplishes semi-regulated learning, yet additionally effectively further develops the discovery pace of malevolent Android APPs, as indicated by the outcomes.

Sun et al. (2021) [22] fostered a profound learning procedure for recognizing malware dependent on information procured from a web crawler that gave solicitations to both harmless and hurtful areas on the Internet in a methodical way. We utilized the removed significant level organization traffic ascribes to prepare a profound neural organization to recognize harmless and pernicious streams in the wake of utilizing cycles to parcel the organization streams and concentrate highlights. At long last, they evaluated our malware location strategy utilizing an assortment of models, like accuracy, review, and f1 score. The acquired f1 score of 0.924 approved the discovery plan's general exhibition.

3 Methodology

Figure 2 depicts the proposed block diagram of the system in which the initial dataset is collected from the Malimg dataset repository where over 25 family variants of malware are listed, then it is passed for pre-processing where noise and other anomalies are removed using standardization. Then pre-processed bit image is passed to the feature extraction process where essential features are extracted from the bit images using PCA and passed to dimensionality reduction. Here, with the help of PSO, we select the features and they are given to the classifier. For this, we use CNN classifier for effective binary classification.

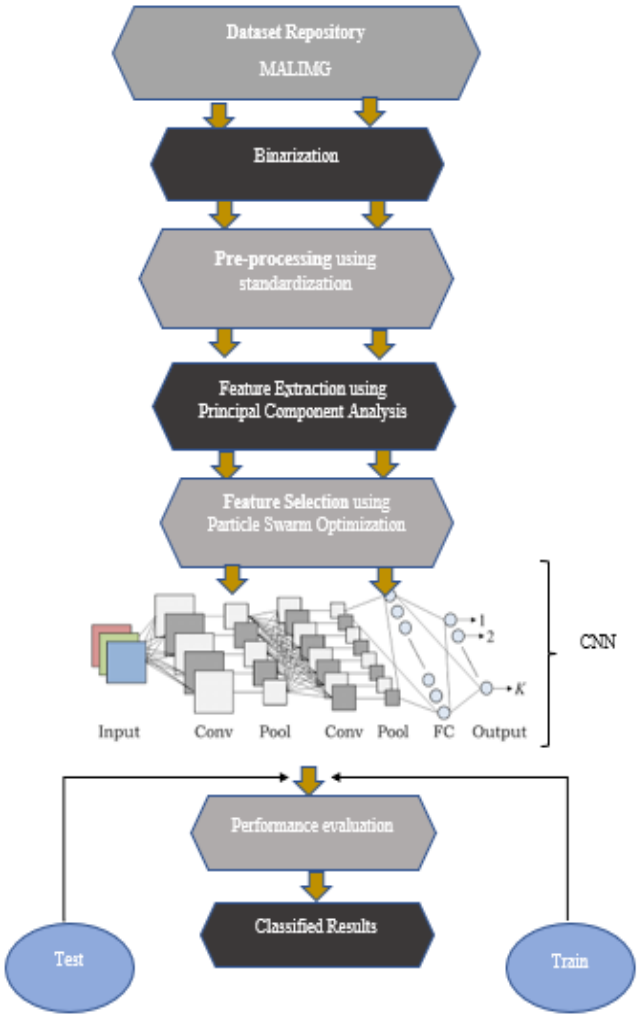

Figure 2: Block Diagram of the Proposed System

3.1 Data Collection

The Malimg dataset contains 9458 malware tests that are ordered into 25 families. The malware tests are not given straightforwardly, but instead as pictures as they show up on circle, which is the critical component of this dataset. The bytes of executable records are inconsequentially moved to skims, which are then deciphered as grayscale picture pixel esteems, like the work in [23, 24]. The dataset's classes

are seriously slanted, true to form: the biggest ('Allaple. A') involves 2949 examples, while the smallest ('Allaple. B') contains just 80 examples. Table 1 shows the Malimg dataset families, and Figure 3 shows Malimg dataset tests.

Table 1: Families of Malimg

No.	Family	Family Name	No. of variants
1	Dialer	Adialer.C	122
2	Backdoor	Agent. FYI	116
3	Worm	Allaple.A	2949
4	Worm	Allaple.L	1591
5	Trojan	Alueron. gen! J	198
6	Worm: Auto IT	Autorun.K	106
7	Trojan	C2Lop.P	146
8	Trojan	C2Lop. gen! G	200
9	Dialer	Diaplatform.B	177
10	Trojan Downloader	Dontovo.A	162
11	Rogue	Fakerean	381
12	Dialer	Instantaccess	431
13	PWS	Lolyda.AA 1	213
14	PWS	Lolyda.AA 2	184
15	PWS	Lolyda.AA 3	123
16	PWS	Lolyda.AT	159
17	Trojan	Malex. gen! J	136
18	Trojan Downloader	Obfuscator.AD	142
19	Backdoor	Rbot! gen	158
20	Trojan	Skintrim.N	80
21	Trojan Downloader	Swizzor. gen! E	128
22	Trojan Downloader	Swizzor. gen! I	132
23	Worm	VB.AT	408
24	Trojan Downloader	Wintrim. BX	97
25	Worm	Yuner.A	800

3.1.1 Binarization

Since these images are in binary format and can't be understood by the DL model, it converts the file that is provided into its binary code format, creates a hex code for the provided binary code of the file and converts it to an image. The Hex dump is then converted to PNG images to feed it inside deep CNN for training [25].

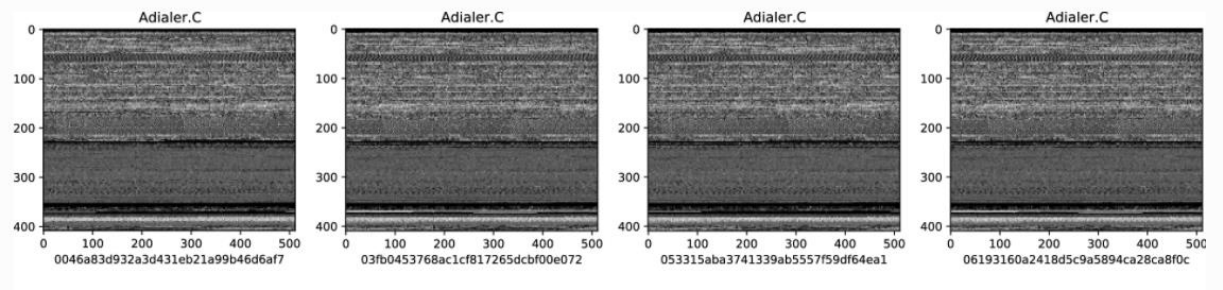

Figure 3: Malimg Samples

3.2 Pre-processing

To meet CNN's feedback information prerequisites, we first preprocess the grayscale picture. At the point when CNN conducts a picture grouping task, it utilizes picture information with similar sizes as the information. The picture information ought to, as a rule, be a similar length and width (the length to width proportion ought to be 1:1). It's for the resulting convolution activity's comfort. Since chief records arrive in an assortment of sizes, grayscale picture sizes change altogether. A gigantic grayscale picture may be pretty much as extensive as 1.04 MB (2048 1036 pixels), yet somewhat one is just 120 KB (512 472 pixels). Accordingly, all grayscale photographs should be standardized. It utilizes the four closest pixels esteems in the first picture to decide a virtual pixel worth of the objective picture, which accomplishes preferable impact over the closest neighbour insertion. Additionally, the standardized size of the grayscale picture is a hyperparameter, which mirrors the compromise between arrangement exactness and estimation cost. The bigger the standardized picture size, the more extravagant the data got by the CNN input; then, at that point, with a more intricate organization structure, a better identification result will be achieved. However, the relating cost is longer tedious for network preparation. To this end, we at long last pick 64 as the standardized size of grayscale pictures [26].

3.3 Feature Extraction

PCA is a dimensionality reduction technique that detects relevant correlations in our data, modifies existing data based on these relationships, and then quantifies the importance of these relationships so that we can keep the most essential ones and discard the rest. (Figure 4).

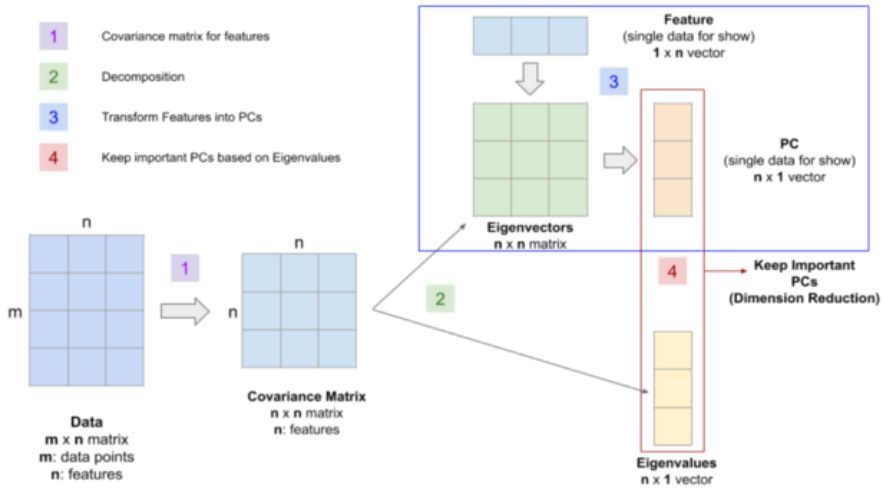

Figure 4: Process of PCA

It involves focusing out at least one of the most vulnerable primary parts, bringing about a lower-dimensional projection of the crude element information with the greatest information fluctuation saved. The symmetrical, straight projection method is utilized to diminish the dimensionality of the picture. The PCA activity can be characterized as follows without forfeiting consensus.

$$Y = XC \quad (1)$$

The projected information grid that involves P essential parts of X with P N is Y Rsxp. Discovering the projection network C Rnxp, which is equivalent to finding the eigenvectors of the covariance lattice of X, or tackling a solitary worth deterioration (SVD) issue for X [27], is the key.

$$X = E\Sigma V^T \quad (2)$$

where ∈ ℝs×s and V ∈ ℝNXN are symmetrical lattices for X's segment and column spaces, separately, and is a slanting grid having the solitary qualities, m, for n = 0, N 1 non-progressively lying along the askew. The projection framework C can be created from the principal P segments of C utilizing, as shown [19].

$$V = [V1,\ldots\ldots\ldots, Vn] \quad (3)$$

where $V_n \in \mathbb{R}^N$ cn = vn is the n^{th} right solitary vector of X the particular qualities in above Eq. are, indeed, the standard deviations of Y along with the essential headings in the space crossed by the segments of C [28]. Subsequently, the difference of X projection along the n^{th} rule part bearing is 2n s. Change is believed to be an estimation of the measure of data and it adds to the information portrayal. One approach to check this is to take a gander at the chief part's total clarified difference proportion, which is given as.

$$R_{ccv} = \frac{\sum_{n=1}^{P} \lambda_n^2}{\sum_{n=1}^{N} \lambda_n^2} \quad (4)$$

3.4 Feature Selection

PSO is a populace based developmental streamlining approach (called a multitude). The current position, speed, and ideal situation of every molecule in the populace are large factors. The particles in the multitude move around in the inquiry space with speed, directed by their own previous best positions and the multitude's most popular positions. At the point when better destinations are found, they are utilized to direct the multitude's future movements. The methodology is rehashed until the best multitude of destinations is distinguished [29, 30].

The molecule boundaries in an n-dimensional hunt space are addressed by n-dimensional vectors. Xi = (x1, x2, x3,.....xn) and Vi = (v1, v2, v3,.....vn) are the position and speed of the t-th molecule, separately. The position and speed of the particles are refreshed by the emphasis during the hunt interaction.

$$V_i(t+1) = \&w \times V_i(t) + c_1 \times rand\ 0 \times \left(P_{pbest} - V_i(t)\right) + c_2 \times rand\ () \times \left(P_{gbest} - V_i(t)\right),$$
$$X_i(t+1) = \&X_i(t) + V_i(t+1), \quad (5)$$

3.5 Classification

A Convolution Neural Network (CNN) (Figure 5) is a feed-forward neural organization enlivened by the association of creature visual cortex [31]. For picture grouping issues, CNN is the present status of-the-craftsmanship neural organization plan. CNN is comprised of neurons with learnable inclinations and loads. The accompanying three parts make up most CNNs [32].

Convolutional layers: These layers successively play out a bunch of convolution tasks (direct sifting) on the picture. These channels ordinarily separate data from the info picture's edges, tones, and shapes. The channels follow up on picture subregions and execute calculations so that every subregion offers a solitary benefit as a yield. This present layer's yield (suppose x) is generally communicated to a non-straight capacity (called ReLU initiation) with the recipe f(x) = max (0, x).

Pooling layers: This layer is responsible for downsampling (i.e., bringing down the spatial goal of the info layers) the information delivered by convolution layers to save preparing time and permit PC assets to deal with the information's scale. The quantity of learnable boundaries in ongoing layers of the organization is brought down because of pooling. Max pooling is a run of the mill pooling approach that protects the most extreme worth in an area (for instance, 2x2 non-covering information pieces) and disposes of the rest.

Fully connected layers: This layer orders the consequences of the convolution and pooling layers. This current layer's neurons are connected to those in the earlier layer. This layer is normally trailed by a Dropout layer, which further develops the model's speculation capacity by forestalling overfitting, a common issue in the profound learning region [33, 34].

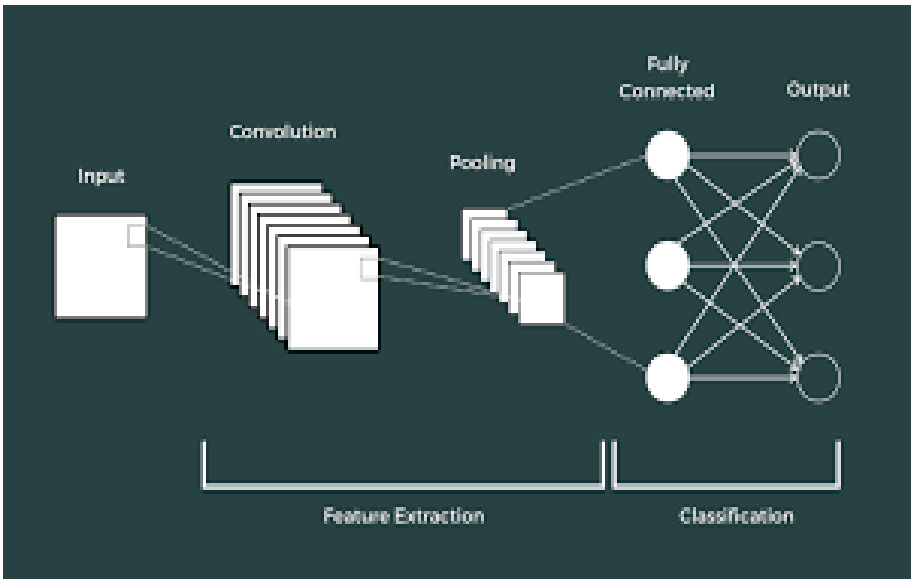

Figure 5: CNN Architecture

4 Performance Analysis

The model is executed utilizing the Python programming language utilizing Intel Core i5 ninth Gen processor, 8 GB/512 GB SSD/Windows 10 Home/4 GB NVIDIA Geforce GTX 1650 Graphics/144 Hz equipment specs. The Google Collab stage is utilized to carry out the model. We estimated exactness, affectability, explicitness, accuracy, review, f1-score, TPR, FPR, and recognition rate to assess our model. Likewise, we evaluated our model over different techniques, for example, VGG16, VGG19, Densenet, Alexnet, Ensemble learning based on the Malimg dataset. Table 2 portray a similar examination of Accuracy, affectability, particularity over different models. Figure 6 (a, b, c) portray a graphical portrayal of precision, affectability and particularity of different models. Table 3 depicts a similar examination of precision, review, f1-score over different models. Figure 7 (a, b, c) shows a graphical portrayal of precision, review, f1-score over different models.

Table 2: Comparative Analysis of Accuracy, Sensitivity, Specificity

Models	Accuracy	Sensitivity	Specificity
VGG16	83	95	97
VGG19	86	93	98
Densenet	93	94	92
Alexnent	81	89	92
Ensemble Learning	87	90	96
D-WARE (Our)	**96**	**98**	**97**

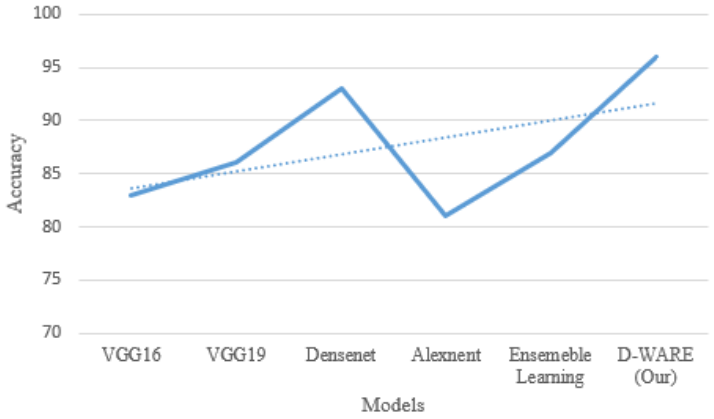

Figure 6 a: Models vs Accuracy

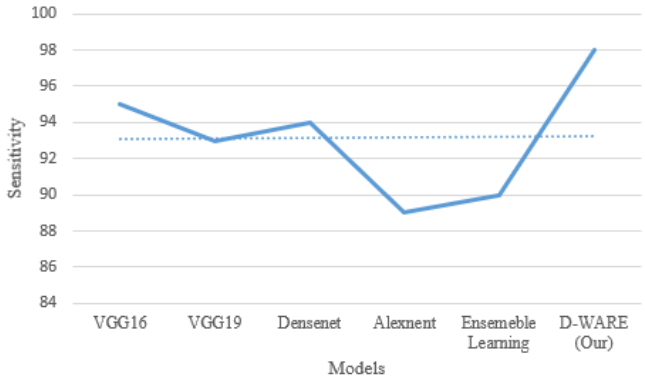

Figure 6 b: Models vs Sensitivity

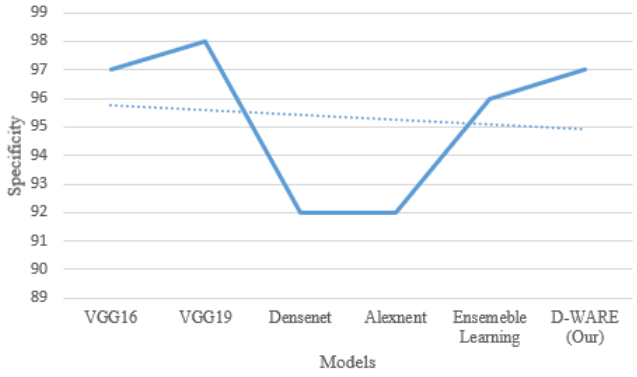

Figure 6 c: Models vs Specificity

Table 3: Comparative Analysis of Precision, Recall, F1-score

Models	Precision	Recall	F1-score
VGG16	0.87	0.93	0.97
VGG19	0.93	0.97	0.98
Desnenet	0.95	0.97	0.94
Alexnet	0.89	0.93	0.94
Ensemble learning	0.92	0.94	0.96
D-WARE (Ours)	**0.96**	**0.91**	**0.98**

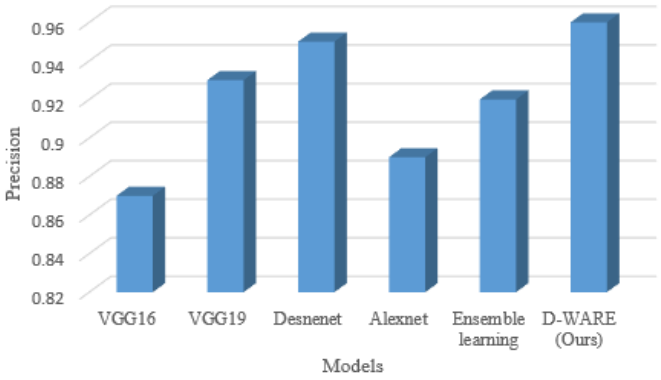

Figure 7 a: Models vs Precision

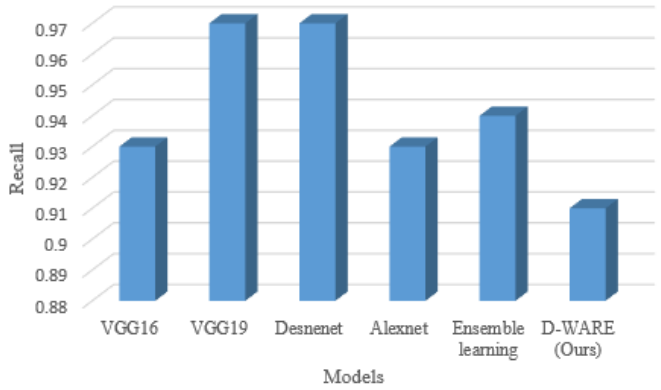

Figure 7 b: Models vs Recall

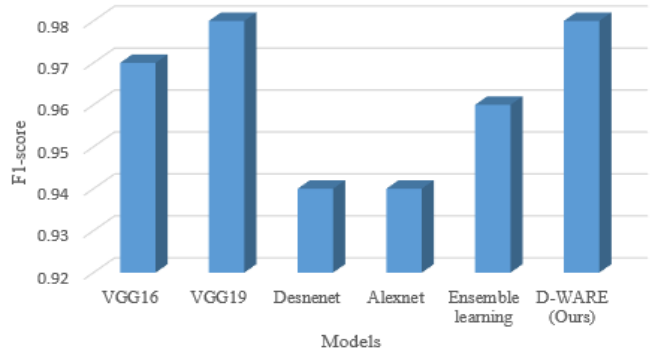

Figure 7 c: Models vs F1-score

Figure 8 shows the graphical representation of True Positivity Rate (TPR), False Positive Rate (FPR) of various models. Figure 9 depict the detection rate in which our model outperforms other models by 97%. Table 4 depicts the comparison analysis of various research analyses with our analysis.

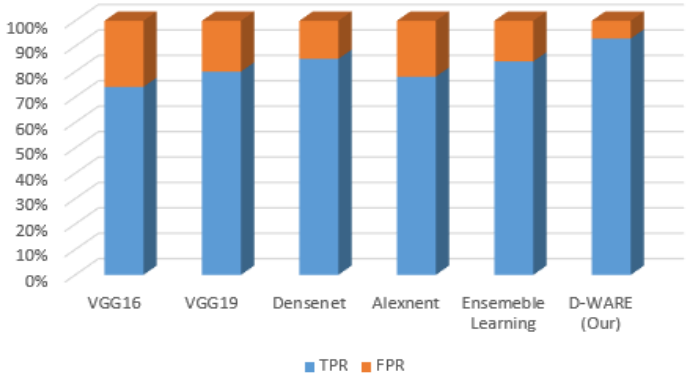

Figure 8: Models vs TPR, FPR

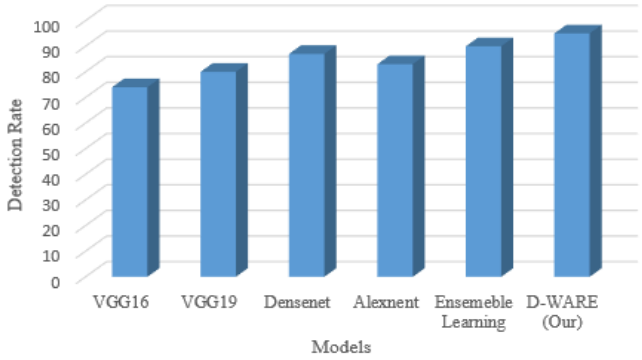

Figure 9: Models vs Detection Rate

Table 4: Comparative Analysis of the Summary of Research Work

Authors	Accuracy
Shukla et al. (2019)	94
Marin et al. (2019)	90
Darem et al. (2021)	92
Poonguzhali et al. (2019)	95
Feng et al. (2020)	95.2
Sun et al. (2021)	92.4
D-ware (Ours)	**96**

5 Conclusion

From this paper, we get to understand the importance of securing the device from malicious attacks. The world is now focusing on improving security in various ways from preventing data leaks. DL brings such an effective environment for building such an efficient model for detecting malware at a higher rate. In this paper, we have so far come across the stages of detecting the malware using the Malimg dataset. Finally classified using CNN and compared with other models over various measures in which we obtained 96% accuracy than other models.

References

[1] Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M.J., Levi, M., & Savage, S. (2013). Measuring the cost of cybercrime. In The economics of information security and privacy, 265-300. Springer, Berlin, Heidelberg.

[2] Li, B., Roundy, K., Gates, C., & Vorobeychik, Y. (2017). Large-scale identification of malicious singleton files. *In Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 227-238.

[3] Alazab, M., Venkataraman, S., & Watters, P. (2010). Towards understanding malware behaviour by the extraction of API calls. *In IEEE second cybercrime and trustworthy computing workshop*, 52-59.

[4] Tang, M., Alazab, M., & Luo, Y. (2017). Big data for cybersecurity: Vulnerability disclosure trends and dependencies. *IEEE Transactions on Big Data*, *5*(3), 317-329.

[5] Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2010). Zero-day malware detection based on supervised learning algorithms of API call signatures. *Proceedings of the Ninth Australasian Data Mining Conference*, *121*, 171-182.

[6] Alazab, M., Venkatraman, S., Watters, P., Alazab, M., & Alazab, A. (2011). Cybercrime: the case of obfuscated malware. *In Global security, safety and sustainability & e-Democracy*, 204-211. Springer, Berlin, Heidelberg.

[7] Alazab, M. (2015). Profiling and classifying the behavior of malicious codes. *Journal of Systems and Software*, *100*, 91-102.

[8] Huda, S., Abawajy, J., Alazab, M., Abdollalihian, M., Islam, R., & Yearwood, J. (2016). Hybrids of support vector machine wrapper and filterbased framework for malware detection. *Future Generation Computer Systems*, *55*, 376-390.

[9] Raff, E., Sylvester, J., & Nicholas, C. (2017). Learning the pe header, malware detection with minimal domain knowledge. *In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 121-132.

[10] Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., & Van Steen, M. (2012). Prudent practices for designing malware experiments: Status quo and outlook. *In IEEE symposium on security and privacy*, 65-79.

[11] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C.K. (2018). Malware detection by eating a whole exe. *In Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

[12] Krčál, M., Švec, O., Bálek, M., & Jašek, O. (2018). Deep convolutional malware classifiers can learn from raw executables and labels only.

[13] Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *computers & security*, *77*, 578-594.

[14] Anderson, H.S., Kharkar, A., Filar, B., & Roth, P. (2017). Evading machine learning malware detection. *black Hat*, *2017*.

[15] Verma, R. (2018). Security analytics: Adapting data science for security challenges. *In Proceedings of the fourth ACM international workshop on security and privacy analytics*, 40-41.

[16] Shukla, S., Kolhe, G., PD, S.M., & Rafatirad, S. (2019). Rnn-based classifier to detect stealthy malware using localized features and complex symbolic sequence. *In 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 406-409.

[17] Marín, G., Casas, P., & Capdehourat, G. (2019). Deep in the dark-deep learning-based malware traffic detection without expert knowledge. *In IEEE Security and Privacy Workshops (SPW)*, 36-42.

[18] Darem, A.A., Ghaleb, F.A., Al-Hashmi, A.A., Abawajy, J.H., Alanazi, S.M., & Al-Rezami, A.Y. (2021). An adaptive behavioral-based incremental batch learning malware variants

detection model using concept drift detection and sequential deep learning. *IEEE Access*, *9*, 97180-97196.

[19] Poonguzhali, N.P., Rajakamalam, T., Uma, S., & Manju, R. (2019). Identification of malware using CNN and bio-inspired technique. *In IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1-5.

[20] Feng, J., Shen, L., Chen, Z., Wang, Y., & Li, H. (2020). A two-layer deep learning method for android malware detection using network traffic. *IEEE Access*, *8*, 125786-125796.

[21] Sun, Y., Chong, N.S., & Ochiai, H. (2021). Network Flows-Based Malware Detection Using A Combined Approach of Crawling and Deep Learning. *In ICC IEEE International Conference on Communications*, 1-6.

[22] Marastoni, N., Giacobazzi, R., & Dalla Preda, M. (2018). A deep learning approach to program similarity. *In Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis*, 26-35.

[23] Marastoni, N., Giacobazzi, R., & Dalla Preda, M. (2021). Data augmentation and transfer learning to classify malware images in a deep learning context. *Journal of Computer Virology and Hacking Techniques*, *17*(4), 279-297.

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

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

[26] Zhang, J. (2019). Machine learning with feature selection using principal component analysis for malware detection: a case study. *arXiv preprint arXiv:1902.03639*.

[27] Mahindru, A., & Sangal, A.L. (2019). Deepdroid: feature selection approach to detect android malware using deep learning. *In IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 16-19.

[28] Kennedy, J. (2011). Particle swarm optimization. Encyclopedia of machine learning. *Springer*, *760*-766.

[29] Xu, Y., Wu, C., Zheng, K., Wang, X., Niu, X., & Lu, T. (2017). Computing adaptive feature weights with PSO to improve Android malware detection. *Security and Communication Networks*, *2017*.

[30] "Convolutional neural network," https://en.wikipedia.org/wiki/ Convolutional neural network, 2017

[31] "Intro to convolutional neural networks," https://www.tensorflow.org/ tutorials/layers, 2017.

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

[33] Sajjad, S., & Jiana, B. (2020). The use of Convolutional Neural Network for Malware Classification. *In IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, 1136-1140.

[34] Rafique, M.F., Ali, M., Qureshi, A.S., Khan, A., & Mirza, A.M. (2019). Malware classification using deep learning-based feature extraction and wrapper-based feature selection technique. *arXiv preprint arXiv:1910.10958*.

[35] Ducau, F.N., Rudd, E.M., Heppner, T.M., Long, A., & Berlin, K. (2019). Automatic malware description via attribute tagging and similarity embedding. *arXiv preprint arXiv:1905.06262*.

[36] Bakhtina, M., & Matulevicius, R. (2022). Information Security Risks Analysis and Assessment in the Passenger-Autonomous Vehicle Interaction. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 13*(1), 87-111.