JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452

AVMCT: API Calls Visualization based Malware Classification using Transfer Learning

[1]Manish Goyal,
1`Department of Computer Science and Engineering, I K Gujral Punjab Technical University, Kapurthala,Punjab,India.Email-id: er.manishgoyal.ghudda@gmail.com

2Raman Kumar
2Department of Computer Science and Engineering, I K Gujral Punjab Technical University, Kapurthala, Punjab,India.Email-id: er.ramankumar@aol.in

Received 2022 March 15; **Revised** 2022 April 20; **Accepted** 2022 May 10.

ABSTRACT: The exponential growth of the internet and high-speed data transmission has also increased the security threat of data.The antivirus companies are providing security to this data. Cybercriminals are in continuous efforts to break security barriers to steal sensitive information and to have unauthorized access or corrupt the victim's system. There is a never-ending cycle between antivirus companies and cybercriminals. There are two ways to detect malware by using static analysis and dynamic analysis. Although static analysis provides fast results,zero-day malware can't be detected as there is a predefined set of signatures in this technique. By using obfuscation techniques malware writers can evade this technique while in dynamic analysis malware detection is based on malware behavior. So, dynamic analysis is capable of detecting new and unseen malware. Machine learning and deep learning techniques are quite effective in the classification of malware on the extracted feature set by using static or dynamic analysis. In a recent study, the malware classification is performed by using transfer learning inConvolution Neural Network (CNN) architectures based on API Call visualization. API Call visualization means converting API Calls in the form of images to detect patterns of different families of malware. After converting API Callimages,the transfer learning is performed on two customized CNN models to enhance feature vectors and made a combined set of feature vectors. The results of thisframework are compared with pre-trained models like VGG-16, ResNet-50 and AlexNet which shows that our suggested approach outperforms pre-trained models.
KEYWORDS: CNN, Deep Learning, Transfer Learning, Malware classification, API Calls

1. INTRODUCTION

The internet and recent technological advancements in computer systems have made the human lifestyle easier and more convenient. These days everything could be done on the Internet, namely socialization, financial transactions, and the assessment of human bodily changes, among other things. All of these improvements encourage cyber thieves to conduct attacks online rather than physically. Malware threats have become one of the most serious dangers to Internet security in recent years. According to McAfee's study 2021[1], there has been a massive growth in malware production recently, with an average of 588 cyber threats each minute. Anti-malware companies are trying to capture this malware. But to avoid detection, malware writers are frequently mutating malware into variants. Also, many tools can build malware in very less time. As a result, malware classification has become a very essential study area. Anti-malware solutions are used to keep malware from infiltrating the system. In this scenario, anti-virus companies must classify malware[2].

Static analysis or dynamic analysis are the most used methods for detecting malware[3]. In general, static and dynamic feature extraction-based algorithms are used to meet the issue of malware classification[4]. The static malware classification method looks for registered signatures in files[5]–[8]. Malware signatures are used to recognize malicious activities using this approach, which is based on virus data produced without the malware being executed. However, when it comes to detecting zero-day malware, signature-based malware detection is ineffective. Dynamic analysis, on the other hand, executes the malware sample in anisolated environment[9]–[13]. So, the behavior of malware is analyzed in this case. So, this method is capable of detecting both disguised and fresh malware.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452
Recently, Deep Learning (DL), a subfield of artificial intelligence, has attained remarkable performance in various fields[14]. Traditional machine learning methods are not enough to detect all types of malware. The working of deep learning techniquesis different from machine learning techniques hence can prove beneficial in the detection of malware samples[15].

Transfer learning refers to the learning outcome of one CNN model to repurposing another CNN model. The CNN model used in transfer learning can be pre-trained models or customized models. In the present paper, we have combined feature vectors of two CNN models. The proposed model is compared with pre-trained models like VGG-16, ResNet-50 and DenseNet-50. The results show that AVMCToutperformed existing models. The significantcontributions of this paper are:

1. Collection of malware samples from various websites and analyzing their behavior in an isolated environment of Cuckoo Sandbox.
2. A data set of API Call Images has been created.
3. The transfer learning of two CNN models is used on this data set.
4. Comparison of the proposed model is done with previously existing models like VGG-16, ResNet and AlexNet.

The organization of the present study is as Section 1presents in the introduction. It includes the introduction of malwareand present work. A literature review of related work is presented in Section 2. It includes the work done by various authors in the area of malware detection. Section 3 presentsthe proposed work. It includes the introduction of a dataset, architecture of proposed work and background of deep learning. Section 4 presents implementation details. This section includes parameters used for implementation. Section 5 includes evaluation metrics while section 6 includes results and discussion. At last, the conclusion of the paper is added in Section 7.

2. RELATED WORK

In the initial phase malware were developed just to make prank but with the evolution of time and the growth of the internet, it has taken a serious form. So, it is necessary to detect malware at an early stage. The work done by various researchers in the field of malware detection is listed in this section.

Malware is becoming more prevalent in terms of volume, type, and intelligence. Different deep learning models are being developed by researchers to detect and classify malware. IMCFN technique based on malware image has been developed by Vasan et al.[16] using CNN Fine-tuned deep learning model. This method transforms the malware binaries into colored images which are further passed to Fine-tuned CNN model to classify the malware families. This method works well for detecting hidden code[16]. They have used the MalImg dataset [17]for their research.One more image-based technique (IMCEC) has been developed by Vasan et al.[18] to classify the malware using an ensemble of different CNN models. They considered that deeper architecture based on different CNN architectures shows different semantic representations of malware images. This model achieves 99% accuracy over unpacked malware with a low false alarm rate. This model takes 1.18 seconds to detect a new sample of malware[18]. Another fine-tuned CNN-based model has been developed by W. El-Shafai et al.[19]to classify malware that uses the advantages of both fine-tuning and transfer-learning with simple feature extraction techniques. This method has a framework of eight different FT CNN models including AlexNet, Places365-GoogleNet, VGG16, Inception-V3, DarkNet-53, MobileNet-V2, DenseNet-201, and ResNet-50. This technique has been tested on the benchmark MalImg dataset[17]. This technique has achieved an accuracy of 99.97%[19].G.Xiao et al.[20]showed that combining deep CNN with an entropy graph improves malware classification. This model contains an automated feature extraction technique. For this purpose, the authors firstly visualize binaries of malware based on entropy, then CNN was implemented to extract patterns and finally SVM was implemented on extracted features to classify malware.

Asam et al. [2] proposed two frameworks namely DFS-MC and DBFS-MC. In the case of DFS-MC, customized CNN models are used to extract features and are provided to SVM for malware classification, whereas in DBFS-MC customized CNN model is made by combining two pre-trained networks namely ResNet-18 and DenseNet -201. The enhanced features are provided to SVM. The MalImg dataset is used for research. Furthermore, a hybrid model classifies the malware by integrating two pre-trained models AlexNet and Resnet -152 Net that is based on Transfer Learning. This model has been tested on the MalImg dataset and 97.78% accuracy is achieved[15]. Another study classifies malware families using a DLMD strategy based on static

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452
methodologies. The DLMD approach classifies malware families by using byte and ASM files combination extraction of feature extraction. Firstly, the authors used two CNN models to extract features from byte data. Afterward, opcodes are used for wrapper class to extract distinctive and crucial features. The feature spaces are combined and made into hybrid feature sets. Then ANN is implemented on this feature vector [21]. Kalash et al. [22]used the CNN model to detect malware. Their method was tested on MalImg and Microsoft datasets. They firstly converted malware binaries into images and provided these images to CNN as grayscale images. The accuracy achieved by this method is 99.175% and 98.52 % on MalImg and Microsoft datasets respectively[22].
As observed from related work done by various authors, we have noted that traditionally static analysis was done to detect malware but that was incapable to detect new malware. Then research was shifted to machine learning classifiers and then towards deep learning schemes. Nowadays, research has taken a new direction in which a hybrid feature vector is used by combining feature spaces of two or more CNN models. Due to this, results have increased extensively. But the major problem lies in the dataset used by researchers is based on the visualization of malware binary files like MalImg [17]and Big Data[23].These data sets are based on images created from binaries of malware and are prone to obfuscation. In this paper, we have worked on a combination of CNNmodels to make a hybrid feature vector on the API Call dataset to cover the research gap of previous research.

3. PROPOSED WORK

Effective detection of malware families is necessary as soon as possible, to minimize the damage done by malware. Rather than employing traditional static features to detect malware or usinga machine or deep learning techniques we have used a hybrid of two CNN models to detect malware. In this paper hybrid deep learning model is introduced which can classify malware families efficiently using visualization of API calls. For this work firstly dataset is created.
To create the dataset the malware samples are picked from various websites like VirusTotal[24], VirusShare[25] and MalShare[26]. Then these malware samples were executed on an isolated environment of Cuckoo Sandbox[27] and 1482 API Calls were extracted from Cuckoo reports. Figure 1 depicts the process of extraction of API Calls.

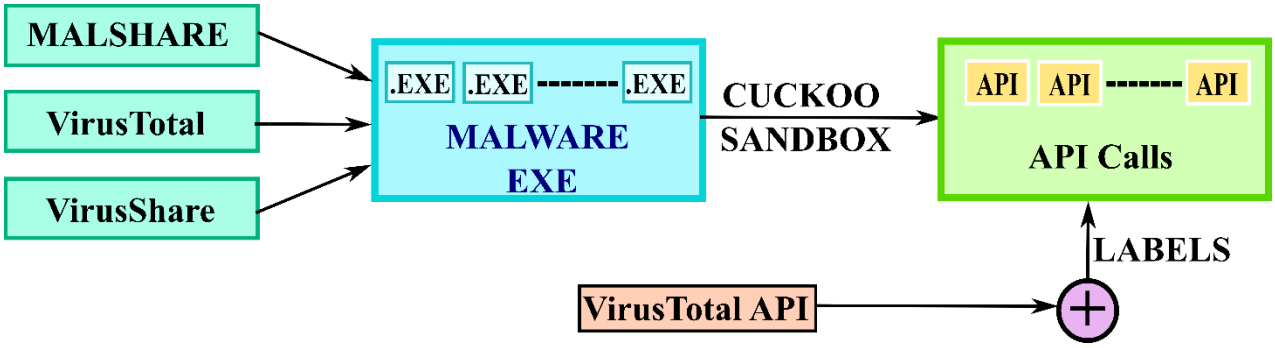

Figure 1:Procedure to extract API Calls

The dataset consists of 15217 malware samples from 14 malware families which are listed in Figure 2.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452

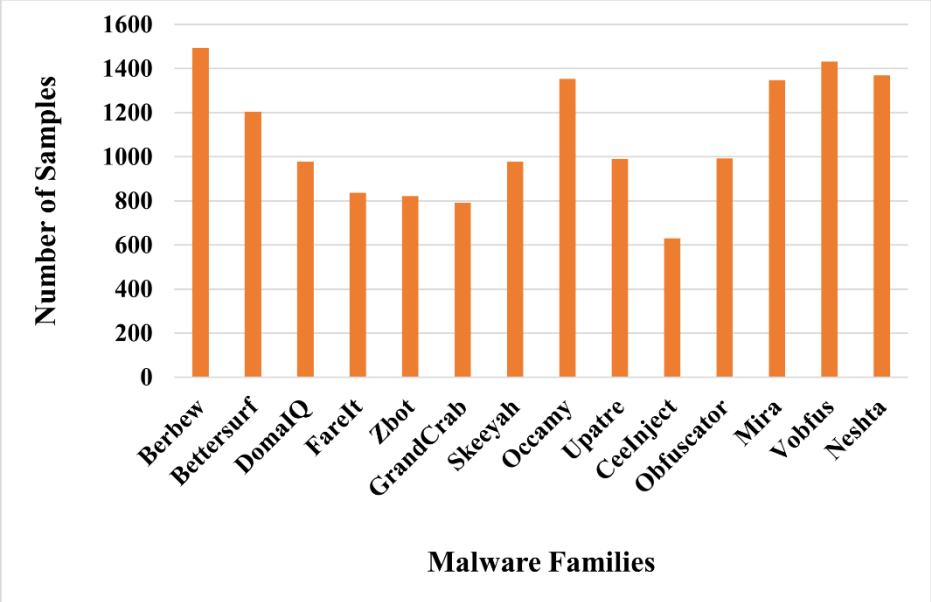

Figure 2: Number of malware samples of each family

3.1. Visualization of API Calls into Images

Application Programming Interface (API) calls allows two software's to talk as it acts as a communicator with the operating system[28]. When applications run on the Windows platform, these applications need an interface to talk to the operating system. API Calls offer this service as whenever any work has to be done by a process, calls are made to the operating system, which is known as API Calls. Consider an example when a process has to delete a file, DeleteFileA,API Call is initiated [29]. As a result, examining these API Calls reveals crucial information about the application's activity.

Visualization of API Calls means converting API Calls into a grayscale image. As deep learning provides the best results on images so there arises a need to convert API Calls into images. As 1489 API Calls were extracted for each sample of malware maximum size of the image which can be obtained is 38 × 39. The distinct API Calls extracted are 264. Hence are indexed from 0 to 263. The value of the pixel in the grayscale image lies between 0-255. To convert the API Calls value into 0-255 each value of the API Call is divided by the maximum value and multiplied by 255. Then 15217 samples of 1482 size are reshaped into 15217 × 39 × 38. The visualization of API Calls of different malware families is shown in Figure 3.

To reshape data from one dimension to two dimensions, for one sample, firstly, the first 38 values of linear data are taken and the next 38 values are added in the 2^{nd} row, the next 38 values are added in the 3^{rd} row and so on. By adopting this procedure, we converted one linear dataset of 1482 feature vectors into two-dimensional data with 39 rows and 38 columns. Then the grayscale images are plotted on this dataset.

While plotting it is observed that the images of different classes show a different pattern and that of the same class provides the same pattern. It can be seen from these images that there are different patterns in each malware family. On behalf of these patterns, deep learning can be implemented.

3.2. Proposed Model of AVMCT

In the proposed model two customized CNN models are combined to utilize their features and to combine features to enlarge the feature vector. In the aforementioned AVMCT technique, we have used a filter size of 5 × 5 in one CNN model and in another CNN model we have used a filter size of 3 × 3 with strides. These different kinds of features are extracted from both CNN models and both kinds of features are combined. The overview of the architecture used for AVMCT is shown in Figure 4.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452

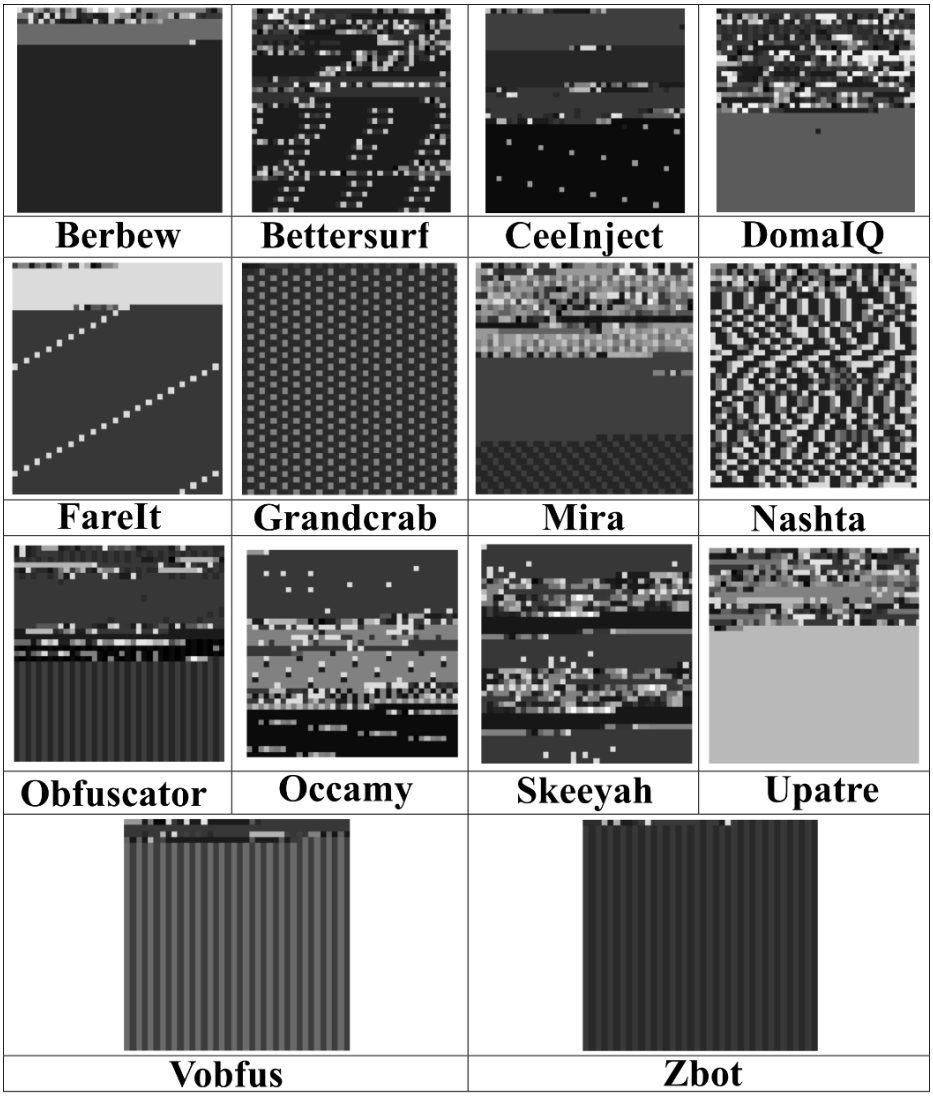

Figure 3: API Call visualization from malware families

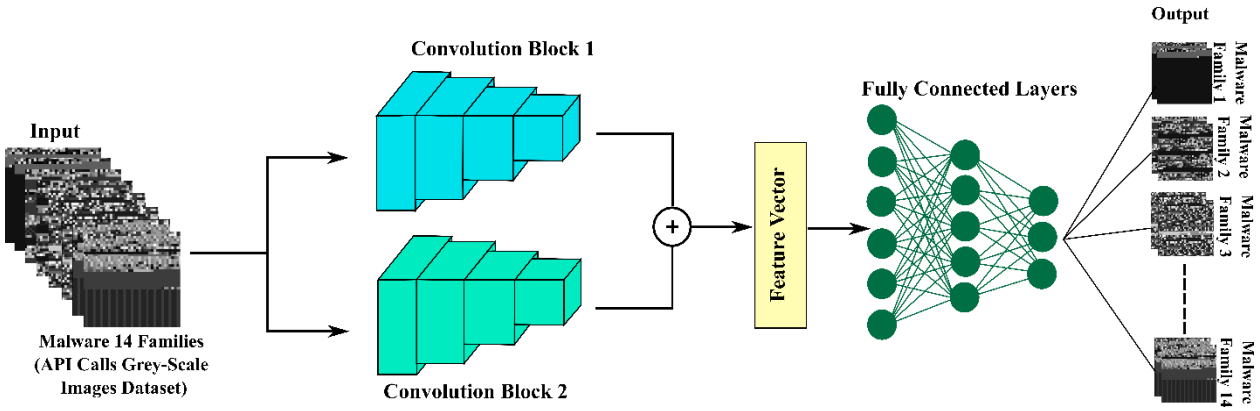

Figure 4: The overview of the architecture of AVMCT

As shown in Figure 4 the API Calls extracted from the dataset are organized into a 2D array then the values of API Calls are scaled from 0 to 255. Then this 2D array is converted into grayscale. After that color mapping is

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452
applied to this image so that we can have a clear distinction of patterns. Afterward, these two customized CNN models were combined to make a hybrid feature vector. Then on this combined feature vector fully connected layers are implemented. The algorithm for AVMCT is given in Algorithm 1.

Algorithm 1: AVMCT

Input: API Image Dataset (API_Img)

Output: Trained Data

1. $for\ each\ X\ \in API_{Img}$
2. $Y \leftarrow X[^{'}Label^{'}]$
3. $X \leftarrow X.drop([^{'}Label^{'}])$
4. $X_{train}, X_{test}, Y_{train}, Y_{test} \leftarrow train\_test\_split(\ X, Y, 0.7, random_{state} = 4)$

#70% data is selected as training while 30% data is selected as testing

5. $Set\ batch_{size} \leftarrow 64, epochs \leftarrow 20$
6. Convert labels into categorical data

   $Y_{train_{cat}} \leftarrow np_{utils}.to_{categorical\ (Y_{train})}$

   $Y_{test_{cat}} \leftarrow np_{utils}.to_{categorical\ (Y_{test})}$
7. $Implement\ CNN\ model\ 1\ by\ applying\ convolution\ block$
8. $Implement\ CNN\ model\ 2\ by\ applying\ convolution\ block$
9. $Implement\ concatenate\ layer$
10. $Apply\ fully\ connected\ layers$
11. $Validate\ the\ data$

3.3. Background of Deep Learning used in AVMCT

Different deep learning models are used in the proposed AVMCT technique to improve the input feature representation. CNN is a sort of artificial neural network that operates on the notion of local connection. As a result, they can take advantage of local correlation by guaranteeing that neurons in neighboring layers are connected locally. CNN's foundation is made up of convolutional layers. In convolution layers filters of fixed sizes are used these filters convolve across the whole image and thus extract features and make a new output matrix. Kernels are automatically evolved during training using the backpropagation technique. As a result, for each place in the input matrix, the output is the result of the convolved kernel.

The major steps in any generic CNN architecture are as follows[30], [31]

1) Apply convolution layers to extract features

2) Apply pool or activation or batch normalization layers to fasten the training process

3) Apply fully connected layers to train the model

Another key element in CNN architecture is pooling. Min, max, and average are all different variations of the pooling layer.Pooling layers are used to reduce the size of the dataset by choosing minimum, maximum and average values chosen out of the window frame used for a min, max and average pooling respectively[32]. To avoid much loss of data pool size is generally taken as 2 × 2. Because of its low computational cost and transition invariance, maximum pooling is often employed. The fully connected layer is utilized in the end after stacking many convolutional and pooling layers. The features are taken from several convolutions and pooling layers. These extracted features are mapped to the output by a fully connected layer. This also adds multilayer perceptron capability to the CNN, but instead of working directly on the input, it applies kernels to the extracted features.

4. IMPLEMENTATION DETAILS

While executing samples on Cuckoo Sandbox Ubuntu is installed on the host machine while the guest machine was installed with windows 7. The firewall of the guest machine was turned off so that malicious samples can perform their activities. The implementation of the CNN of the proposed work is done using the Keras library on google Colab by setting up GPU as runtime.

Two distinct versions of CNN are employed as feature extractors in the proposed AVMCTapproach. As a result, a five-layer deep CNN was trained from the ground up on image representation API requests from 14 different

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452
malware classes in this regard. Table 1 shows the parameters that are optimized throughout training. In contrast, Table 2 discusses the design of CNN in greater depth.

Table1: Parameters of CNN

Parameters	Value
Convolution layer	2
Number of epochs	20
Batch_size	64
Number of classes	14

Table 2: List of filters used and summary of CNN models

Layers	Number and size of filters	Input	Output
Parameters of CNN model 1			
Conv 2D	32, 5 × 5	39 × 38 × 1	35 × 34 × 32
Batch Normalization		35 × 34 × 32	35 × 34 × 32
Activation		35 × 34 × 32	35 × 34 × 32
Dropout		35 × 34 × 32	35 × 34 × 32
Flatten		35 × 34 × 32	38080
Parameters of CNN model 2			
Conv 2D	32, 3 × 3 (with strides)	39 × 38 × 1	13 × 12 × 32
Batch Normalization		13 × 12 × 32	13 × 12 × 32
Activation		13 × 12 × 32	13 × 12 × 32
Dropout		13 × 12 × 32	13 × 12 × 32
Flatten		13 × 12 × 32	4992
Concatenation [CNN1, CNN2]			
Batch Normalization		43072	43072
Dense		43072	256
Batch Normalization		256	256
Activation		256	256
Dense		256	128
Dropout		128	128
Dense		128	64
Output		64	14

5. PERFORMANCE EVALUATION MEASURES

Log loss is employed as an assessment measure in the proposed AVMCT approach. The cross-entropy between correct and anticipated labels is known as log loss. Equation 1 shows the precise method for calculating the log loss [21].

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(\rho_{ij}) \qquad (1)$$

In the above equation, the total number of samples of malware is represented by *N*, while *M* represents a number of classes which is 14 in our case. ρ_{ij} represents the probability of ith sample of the jth class. The log function

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452
mentioned in the aforementioned formula is Natural Logarithm. Another measure to evaluate the performance of AVMCT is accuracy which can be defined using equation 2 [21].

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (2)$$

Here, *TP* and *TN* are the number of samples that are correctly classified by model while *FN* and *FP* are samples that are incorrectly classified by classifiers.

6. RESULTS AND DISCUSSION

This section includes the results of the proposed AVMCTalgorithm. Also, the comparison of the purposed AVMCTis done with pre-trained models like ResNet-50, AlexNet and VGG-16. The proposed model was executed 10 times and, on each run, the log loss and accuracy in percentage are shown in Table 3.

Table 3: Performance of proposed AVMCT

Sr. No.	Log loss (in %age)	Accuracy (in %age)
1	0.71	98.31
2	0.58	98.61
3	0.45	98.79
4	0.41	98.96
5	0.40	98.96
6	0.39	98.94
7	0.37	99.03
8	0.32	99.08
9	0.34	99.01
10	0.31	99.16
		0.9888 ± 0.0057

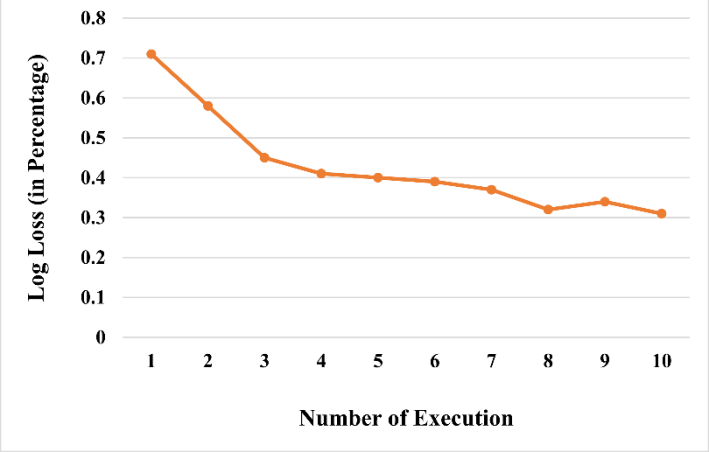

Figure 5: Log loss on each execution

As depicted in Table 3 that the log loss is decreasing and accuracy is almost increasing at each run which states that the model is improving with each run. The execution of CNN takes place it adjusts its weights according to previous output. Therefore, there is a continuous improvement but after a certain point to time validation accuracy becomes high than training accuracy which means the model becomes over-fitted. While training, we continuously monitored the results so that model don't become over-fitted. Therefore, we have taken 20 epochs and executed them 10 times. The graphical representation of log loss and accuracy on each run of epoch forthe proposed work is shown in Figure 5 and Figure 6 respectively.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p.31-41
https://publishoa.com
ISSN: 1309-3452

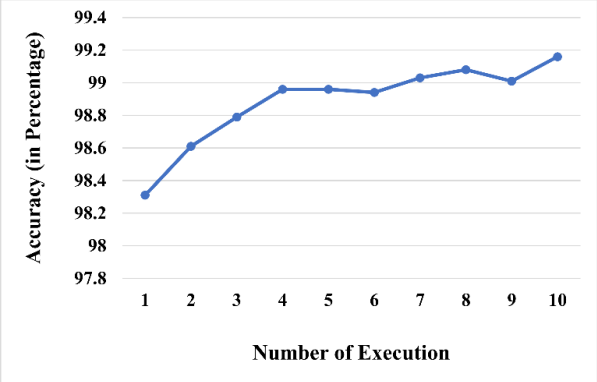

Figure 6: Accuracy of proposed work on execution of each epoch

To check the performance of the model the dataset is fed to pre-trained models like VGG16 [33], Resnet-50 [34] and AlexNet [35]. The accuracy achieved by AVMCT is compared with these pre-trained models is shown in Table 4.

Table 4: The accuracy achieved by various CNN models

CNN Models	Accuracy Achieved (in %age)
VGG16	94
Resnet-50	98.13
Alexnet	98.75
AVMCT (Proposed Work)	98.88

The graphical representation of accuracy achieved by pre-trained CNN models and the proposed hybrid CNN model (AVMCT) is shown in Figure 7.

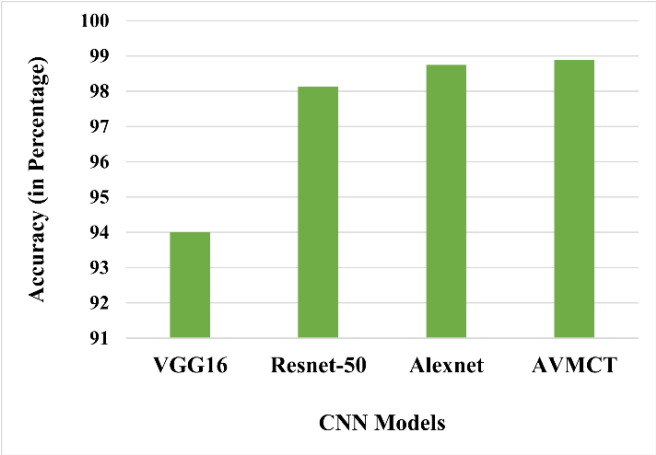

Figure 7:Comparison of Accuracy achieved bya proposed model with pre-trained models

.

As depicted in Table 4 and Figure 7 the accuracy of the proposed AVMCT is higher than pre-trained models. The accuracy achieved by the proposed AVMCT is 4.93% higher than VGG-16, 0.75% higher than ResNet-50 and 0.13% higher than AlexNet.

7. CONCLUSION AND FUTURE SCOPE OF WORK

With the mushroom growth of the internet, malware has also grown tremendously. It has caused security a major concern for cyber security providers.Many antivirus companies have developed signature-based detection methods to detect malware. But malware writers use obfuscation techniques like dead code insertion, insertion

of jump statements or reordering of code to prevent their malware code from being detected. Moreover, there are many tools available in the market with can make new malware and obfuscate previous build malware within a few minutes. This has caused the detection of malware at an early stage to minimize the damage. The machine learning and deep learning techniques are highly effective for malware detection on extracted features of malware. In the present paper, the work is done on transfer learning by combining two customized CNN models. To accomplish this task firstly malware samples of 14 malware familieswere collected from various websites, then these samples were executed in an isolated environment of Cuckoo Sandbox. After that, API Call features were extracted from the JSON file of the cuckoo report. Then these API Calls were converted into images. Afterward, two CNN models were implemented on these API Call images and their features are combined to make a combined feature vector. Then training is provided on this combined feature vector. The proposed work is compared with pre-trained models like VGG-16, ResNet-50 and AlexNet and the results show that our proposed work outperforms the existing techniques. In the future, more features will be extracted apart from API Calls only. Moreover, work can be extended tothe advanced form of malware that can recognize that it is being executed in an isolated environment.

ACKNOWLEDGMENT

The authors wish to thank many anonymous referees for their suggestions to improve the paper. Manish Goyal would like to thank I.K. Gujral Punjab Technical University for offering the Ph.D. course in Computer Science & Engineering and providing support to access the resources for research.

REFERENCES

[1] "McAfee Labs Threats Reports – Threat Research | McAfee." https://www.mcafee.com/enterprise/en-us/threat-center/mcafee-labs/reports.html (accessed Mar. 28, 2022).

[2] M. Asam *et al.*, "Detection of exceptional malware variants using deep boosted feature spaces and machine learning," *Appl. Sci.*, vol. 11, no. 21, p. 10464, 2021.

[3] I. Almomani and A. Khayer, "Android applications scanning: The guide," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, 2019, pp. 1–5.

[4] M. Goyal and R. Kumar, "The Pipeline Process of Signature-based and Behavior-based Malware Detection," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, 2020, pp. 497–502.

[5] C. Yang *et al.*, "DeepMal: maliciousness-Preserving adversarial instruction learning against static malware detection," *Cybersecurity*, vol. 4, no. 1, pp. 1–14, 2021.

[6] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," *ArXiv Prepr. ArXiv13082831*, 2013.

[7] H. V. Nath and B. M. Mehtre, "Static malware analysis using machine learning methods," in *International Conference on Security in Computer Networks and Distributed Systems*, 2014, pp. 440–450.

[8] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *J. Syst. Archit.*, vol. 112, p. 101861, 2021.

[9] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, "Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions," *Appl. Sci.*, vol. 12, no. 1, p. 172, 2021.

[10] E. Amer, I. Zelinka, and S. El-Sappagh, "A Multi-Perspective malware detection approach through behavioral fusion of API call sequence," *Comput. Secur.*, vol. 110, p. 102449, 2021.

[11] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-lady: Deep learning based android malware detection using dynamic features," *J. Internet Serv. Inf. Secur. JISIS*, vol. 11, no. 2, pp. 34–45, 2021.

[12] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *J. Signal Process. Syst.*, vol. 93, no. 2, pp. 265–273, 2021.

[13] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, "Data augmentation based malware detection using convolutional neural networks," *PeerJ Comput. Sci.*, vol. 7, p. e346, 2021.

[14] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, p. 102526, 2020.

[15] Ö. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *Ieee Access*, vol. 9, pp. 87936–87951, 2021.

[16] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, p. 107138, 2020.
[17] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, 2011, pp. 1–7.
[18] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," *Comput. Secur.*, vol. 92, p. 101748, 2020.
[19] W. El-Shafai, I. Almomani, and A. AlKhayer, "Visualized malware multi-classification framework using fine-tuned CNN-based transfer learning models," *Appl. Sci.*, vol. 11, no. 14, p. 6446, 2021.
[20] G. Xiao, J. Li, Y. Chen, and K. Li, "MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks," *J. Parallel Distrib. Comput.*, vol. 141, pp. 49–58, 2020.
[21] M. F. Rafique, M. Ali, A. S. Qureshi, A. Khan, and A. M. Mirza, "Malware classification using deep learning based feature extraction and wrapper based feature selection technique," *ArXiv Prepr. ArXiv191010958*, 2019.
[22] M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, 2018, pp. 1–5.
[23] "Microsoft Malware Classification Challenge (BIG 2015)." https://kaggle.com/competitions/malware-classification (accessed Mar. 28, 2022).
[24] "VirusTotal - Home." https://www.virustotal.com/gui/home/upload (accessed Mar. 29, 2022).
[25] "VirusShare.com." https://virusshare.com/ (accessed Mar. 29, 2022).
[26] "MalShare." https://www.malshare.com/ (accessed Mar. 29, 2022).
[27] "Cuckoo Sandbox - Automated Malware Analysis." https://cuckoosandbox.org/ (accessed Mar. 29, 2022).
[28] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *J. Inf. Secur. Appl.*, vol. 40, pp. 44–51, 2018.
[29] mikben, "DeleteFileA function (fileapi.h) - Win32 apps." https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-deletefilea (accessed Mar. 27, 2022).
[30] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020.
[31] H. H. Aghdam and E. J. Heravi, "Guide to convolutional neural networks," *N. Y. NY Springer*, vol. 10, no. 978–973, p. 51, 2017.
[32] M. Goyal and R. Kumar, "A Survey on Malware Classification Using Machine Learning and Deep Learning," *Int. J. Comput. Netw. Appl.*, vol. 8, no. 6, pp. 758–775, 2021.
[33] A. Zisserman and K. Simonyan, *Very Deep Convolutional Neural Network for Large-Scale Image Recognition*. ICLR, 2015.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.