Robust Malware Detection using Residual Attention Network

Shamika Ganesan*, Vinayakumar Ravi†, Moez Krichen‡, Sowmya V*, Roobaea Alroobaea§, and Soman KP*

*Computational Engineering and Networking (CEN), Amrita School of Engineering
†Amrita Vishwa Vidyapeeth Coimbatore, India. Email: shamikaganesan@gmail.com
‡Center for Artificial Intelligence, Prince Mohammad Bin Fahd University, Khobar, Saudi Arabia.
§Faculty of CSIT, Al-Baha University, Saudi Arabia. and ReDCAD Laboratory, University of Sfax, Tunisia.

Abstract—Recent advancements in Cyber Security have amalgamated the strengths of Artificial Intelligence and Human Intelligence for Intrusion Detection. The colossal increase in the volume of new malware generated everyday and the constant risk of zero day attacks demand research for a robust malware detection system. Significant research has gone into exploring the use of Machine Learning and Convolutional Neural Networks (CNNs). There has been a transition from using Malware byte information for Machine Learning and Deep Learning based methods to using an Image based Intrusion Detection system for better assessment of the malware file. Though CNNs have helped in capturing local features, Attention based mechanisms play a vital role in detecting structural changes in malware. In this paper, we have explored the use of Residual Attention for malware detection and have compared this with the existing CNN based methods and conventional Machine Learning algorithms using GIST features. The proposed method could efficiently focus the 'attention' to precisely those sections of the malware which were significant in being distinguished from benign files, thus reducing the False Positives which is primary from the Accessibility point view in Cyber Security. The method is robust against structural changes in malware and has outperformed traditional malware detection methods by demonstrating an accuracy of 99.25%.

Index Terms—CyberCrime, Cyber Security, Residual Attention.

I. INTRODUCTION

The onset of the fourth industrial revolution (Industry 4.0), approximately hosts 4.54 billion active internet users who meet on the web with umpteen number of information being shared around. With the exponential growth of the Internet of Things (IoT), most of the devices are connected to each other by transmitting zetabytes of data across the globe [1], [2]. Each piece of information might be confidential to the user or the organization concerned [3]. Malware, short for malicious software, exist in different forms and have different ways to execute an attack. Some stay dormant and compromise the system over a period of time, some execute as soon as they enter an endpoint device. According to the Cyber Security Report 2020 published by the National Technology Security Coalition (NTSC) [4], there is an escalation of sophisticated and targeted ransomware attacks with a high level of intelligence involved in the attacks. Therefore, having a generalized mechanism to detect any malware file becomes a challenge.

Intrusion Detection had been approached using Static and Dynamic analysis in the past. Static analysis involves examining the malware without executing the code. Usually Static analysis involves signature identification using disassembly of the malware binary file. Dynamic analysis involves execution of malware in a controlled environment to study its behavior [5]. Hybrid analysis involves gathering information from both Static and Dynamic malware analysis for a more intensive and robust inspection of malware. The flip side is that it is computationally very heavy and resource consuming. Thus, Machine Learning and Deep Learning came in to bridge the gap [6].

Machine Learning models have been used with the features of malware obtained from static, dynamic and hybrid analysis which have shown promising results to detect obfuscated malware [7] [8]. But this approach does not solve the purpose of having to go through tedious feature engineering. Also, if the input features are not rightly picked then the corresponding results fail accordingly. Thus, exploring Deep Learning based techniques had to come to the forefront to avoid explicit feature extraction [9] [10].

Extracting binary information from malware byte files and visualizing them as images has been experimented with for applying Computer Vision techniques in Deep Learning for classification [11] [12]. Such methods facilitate Cyber Security research to explore a new dimension of handling malware data and enable more efficient detection with the help of CNNs and Deep Learning concepts. CNNs are very efficient for extracting local features but to deal with polymorphic malware and zero day attacks, a more generalizable technique is required. Attention mechanisms addressed this issue by not only focusing on the local feature information but also analysing a global representation of the input. Thus, in this paper, we have explored Residual Attention for malware detection. The major contributions of the proposed work is...
TABLE 1: Literature based comparison of existing Malware detection techniques with the Proposed Method.

| Reference | Type of detection | Methodology | Summary |
|-----------|------------------|-------------|---------|
| [15]      | Static Analysis  | Naive Bayes, Support Vector Machines, Decision Trees and their boosted versions. | Boosted Decision Trees gave the best results. Lack of readable descriptions useful for computer-forensic experts. |
| [16]      | Image Processing, Machine Learning | Binary Texture Analysis (GIST features) | Scalable malware classification system due to robustness towards packing strategies. Data intensive method. Can be compromised once the algorithm is known. |
| [20]      | Deep Learning, Image based Techniques | Convolutional Neural Networks (CNN), Transfer Learning | Efficient classification. Requires generalization to tackle structural variations. |
| [22]      | Deep Learning, Image based Techniques | Convolutional Neural Networks (CNN) | Deep learning outperforms Machine Learning methods. CNNs are vulnerable in adversarial environments. |
| [10]      | Deep Learning, Image based Techniques | Convolutional Neural Networks (CNN), Recurrent Neural Networks(RNN) | RNN provides similarity information between time-steps. Trained on a very small dataset; cannot be generalized. |
| Proposed Methodology | Deep Learning, Image based Techniques | Residual Attention with CNN | Captures variations in malware features efficiently to enable accurate classification of Polymorphic malware. Inability to capture variable length inputs; can result in information loss. |

as given below:

- An image based malware detection mechanism was proposed based on a detailed investigation and analysis of Residual Attention networks for malware detection.
- Performance comparison with classical method i.e. GIST with machine learning algorithms.

The remaining sections of the paper has the following organization of content. A study on classical feature engineering methods for malware detection is described in Section 2. Residual Attention is proposed and architecture is discussed in more detail in Section 3. In Section 4, the experimental results are presented, and in Section 5 a discussion on the performance evaluation along with various visualization techniques like Saliency maps, Layer Activation plots, t-distributed stochastic neighbour embedding (t-SNE) plots, heatmaps etc for better understanding of the proposed method is presented and an overall summary of the paper along with a discussion on future scope is provided.

![Fig. 2: Mask(M) and Trunk(T) branches of the Residual Attention unit for input (x).](image)

II. RELATED WORKS

Before the intervention of AI in the domain of Cyber Security, malware analysis and classification was handled using two techniques - Static and Dynamic analysis. In Static analysis, executables have to be unpacked and decrypted before analysis [7]. This process is computationally heavy, time consuming and suffers from code obfuscation. In Dynamic analysis, all characteristics of the malware file might not be observed due to the execution environment constraints. This brought in the intervention of Machine Learning and Deep Learning based techniques. Machine Learning based approaches used features from behavioral analysis obtained from the above stated methods [13] [14]. Due to the complexity involved in behavioral analysis, the next phase of malware classification involved visualizing malware byte files as images using which a texture based analysis using GIST features was carried out by [11] [15] [16]. But these techniques could be easily outmanoeuvred by malicious users who understand the working behind these algorithms. Thus, the intervention of Deep Learning was necessary to pave a pathway to better security systems.

In [17], authors have performed consolidated experiments for comparing GIST based features for classical Machine Learning algorithms viz. Support Vector Machines and k-Nearest Neighbours versus a Deep Learning approach for malware classification and have proven that using Deep Learning based approaches have been advantageous in several ways. There have been researches based on CNNs and several CNN based architectures and Transfer Learning methodologies were proposed over time [18] [19] [20]. At this juncture, the question arose as to how well these algorithms can tackle polymorphic malware and zero-day attacks, since CNNs used in Deep Learning could capture spatial features but it was important to be able to capture patterns in temporal sequences. Thus, there came the introduction of the use of Long-Short Term memory (LSTMs), Gated Recurrent Units (GRUs) and Recurrent Neural Networks (RNNs) for malware classification [21] [10]. During recent times, the use of Attention based mechanisms have started to come to the forefront. Significant work has gone into using different kinds of Attention networks for byte level information as well as converted image malware datasets [22]. In this paper, we explore the use of Residual Attention and compare it with the existing techniques of Texture analysis and CNN based architectures.
A. Attention Networks

CNNs form the building blocks for most of the Computer Vision Deep Learning techniques. An image is just a multi-dimensional array of values (pixels). A linear combination of these pixel values following a certain pattern (convolution with different filters) gives an output which is also an array of values but with useful information regarding the spatial spread of relevant information in the input image enabling efficient classification [23]. Attention Networks are models which decide how much attention needs to be paid to which part of the input. Such region based analysis helps in focusing on the right sections of an image for enabling classification using the right features. Fig.1 explains the significance of region based analysis of malware images. The malicious content in a malware image could be in any section of the image and most of the malware obfuscation occurs due to the lack of a global analysis of the image. Attention networks help in a more generic view of the input images with the help of Bottleneck convolutions and Max-pooling layers to overcome this challenge [24] [25].

B. Residual Attention Networks

Residual Attention networks have multiple Attention units interacting with each other. Each unit consists of two branches - the Mask and the Trunk branch. The Mask branch uses a bottom-up top-down approach to output weight values that would be used to weight the output features produced by the Trunk branch. This also acts as gates for the Trunk branch during backpropagation to ensure that Gradient descent does not stall, and thus the vanishing gradient problem is avoided, as in the case of Highway Networks [26]. The Trunk branch is used to get the features corresponding to each of the input images. Thus, “attention” is paid to those parts of the features calculated by the Trunk branch, which have a higher weight value as per the Mask branch. Here, Mask branch acts as a feature selector. Fig. 2 explains how the Mask(M) and Trunk(T) branches work effectively to produce Attention maps from an image input(x). The output of an Attention module A is given as follows:

\[ A_{j,k}(x) = M_{j,k}(x) \ast T_{j,k}(x) \]  

(1)

During backpropagation, Attention units are robust to noisy labels due to the following update:

\[ \frac{\partial M(x,\alpha)T(x,\beta)}{\partial\beta} = M(x,\alpha)\frac{\partial T(x,\beta)}{\partial\beta} \]  

(2)

where \( \alpha \) are the parameters associated with the Mask branch and \( \beta \) are the parameters associated with the Trunk branch. In [25], the Trunk branch feature processing is done using pre-activation Residual Units.

To avoid the performance drop due to naive stacking of Attention units, authors of [25] have described Attention Residual Learning where the connections between Attention units have identical mappings and is thus the Attention module output is modified to the following:

\[ A_{j,k}(x) = (M_{j,k}(x) + 1) \ast F_{j,k}(x) \]  

(3)

where \( M(x) \) is in the range of [0,1]. With \( M(x) \) approximating to zero, \( A(x) \) takes the value of the original features \( F(x) \).

Fig. 3: Block Diagram for our proposed architecture. The input grayscale image is resized to 96x96 and fed into a 2D convolutional layer followed by Max-pooling. The output from the Attention unit undergoes Average Pooling and is fed to a fully connected layer network with dropouts after each fully-connected layer. The output is finally fed into the classifier to determine whether the image is a malware or benign image.

IV. Dataset and Experimental Design

A. Dataset Description

In this work we have used a Windows Malware dataset which is a binary class dataset with 3012 benign and 3000 malware PNG files of size 256 x 256 dimensions (grayscale). This dataset was created by [27] in their work for Malware detection by converting the malware byte information into images. Every malware binary is computed as an 8-bit unsigned integer vector. The vectors are reshaped to obtain 256x256 matrices which are saved as gray scale images with a range of 0-255, thus giving a visual representation to the malware binaries. For our experiments, we have taken an 80-20 train - test split since there is a balance in the number of malware and benign images and the total number of images in the dataset is limited to around 6,000. The train-test split was carried out using the Model Selection package in the scikit-learn Machine Learning Library\(^1\) in Python, which ensures that the test data is completely unseen for the trained model.

B. Proposed Methodology

In this work, we propose an architecture (as shown in Algorithm 1) containing CNN layers along with Residual Attention Networks to check for correlations between spatial information of malware images and for increasing robustness for polymorphic malware. Our model consists of an input layer which takes in grayscale images, a convolutional layer with 32 filters of kernel size 3x3, one attention unit post which an average pooling of 2x2 with a stride of 2x2 is carried out to

\(^1\)https://scikit-learn.org/stable/
avoid overfitting. This output is fed into the final layer for
classification where the input is classified based on whether it
is a malware file or a benign file. The Attention unit’s Mask
and Trunk branch majorly use three Convolution layers with
32, 64 and 128 filters each after which Residual Learning
happens with the output from the Mask branch and the Trunk
branch. A detailed Block Diagram for our proposed model
is given in Fig. 3. Training was done for 100 epochs with a
batch size of 16 using an Adam Optimizer and a learning rate
of 0.0001. The training was limited to 100 epochs due to the
increase in training loss with an increase in the number of epochs.

Algorithm 1: Residual Attention Model

Input: in_shape, n_class, activation. in_shape is a
tuple, n_class is an integer, activation is a
string.
Output: model
initialization;
Residual Network (in_shape, n_class, activation)
Input ← image
2dConvolution (32 filters, 3x3 kernel, 1x1 stride)
   ← Input
MaxPool( 2x2 size, 2x2 stride, ’same’ padding )
   ← 2dConvolution
Attention ( [32, 64, 128] filters ) ← MaxPool
AvgPool ( 2x2 size, 2x2 stride ) ← Attention
Flatten () ← AvgPool
Dense ( n_class, activation ) ← Flatten
return Dense

V. PERFORMANCE EVALUATION

A. Experimental Results

The experiments were carried out on Google Colaboratory 2
with a GPU based hosted runtime RAM support of 25GB. The
codes were written in python using Keras3 version 2.4.3 with a
backend of TensorFlow 4 version 1.15 for the implementation
of our architecture. After trying different network structures,
we found that different numbers of Residual pre-activation
units and CNNs for initial feature extraction used in the
network give different results. We had majorly experimented
by using the number of Residual pre-activation units in the
range of 1 to 4. The results for the experiments are given
in Table 2. We observed that less number of Residual pre-activation
blocks give better results for malware images due to
their limited complexity. More number of Residual units
results in performance drop by generating more False negatives,
thereby observing lower Recall values and F1-scores. Also,
a drop in Precision values indicate that increasing the
Residual pre-activation units also increase the generated False positives,
thereby indicating a false alarm for a benign file.

This might cause a hindrance to availability of genuine files
to the users if the files are locked for further analysis by the
malware detection system. A higher False negative value poses
a high threat of a system getting compromised, thus allowing
an intrusion to occur.

TABLE 2: Experiments with different numbers of Residual
pre-activation units used in the Residual Attention Network
and the corresponding Accuracy obtained for each experiment.

| No. of Residual Pre-activation Units | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|-------------------------------------|--------------|---------------|-----------|-------------|
| 0                                   | 99.25        | 99.25         | 99.25     | 99.25       |
| 1                                   | 98.92        | 98.92         | 98.92     | 98.92       |
| 4                                   | 95.67        | 95.79         | 95.67     | 95.67       |

TABLE 3: Experimental Results compared with the standard
algorithms existing in literature.

| Proposed Methodology | CNN [19] | GIST + SVM [17] | GIST + KNN [11] | CNN [27] |
|----------------------|----------|-----------------|-----------------|----------|
|                      | 99.25%   | 98.34%          | 97.25%          | 96.61%   | 95.60%   |

Fig. 4: t-SNE plot for the penultimate layer of the proposed
architecture for testing images.

The t-SNE plot for the penultimate layer for obtained results
is shown in Fig.4. The plot gives a spread of the difference
in features which are mapped into a two dimensional plot,
indicating the two different classes of files viz. Malware and
Benign, which are non-linearly separable. The plot gives a
visual explanation regarding the separability of the features
of the images which is efficiently captured by the proposed
architecture.

Saliency map represents the gradient of the predicted outcome
of the model with respect to the initial input features that it receives. Partial derivatives look at local sensitivities
detached from the decision boundary of the classifier [28].
This gives information regarding the regions in an image or its
extracted features which led to it being classified into a certain
class by the model. Thus, a critical analysis based on Saliency
maps for Malware and Benign images has been carried out,
which is shown in Fig. 7(a) and Fig.7 (b). The variations in
the region of interest captured by our proposed architecture
with respect to both the classes is clearly articulated in the
images.

Despite being able to view the variations of the Malware and
Benign classes which has been analysed, to be able to visualize
the parameters learned by the model which enabled efficient
classification is crucial for understanding any architecture.

---

2https://colaboratory.google.com/notebooks/intro.ipynb
3https://keras.io/
4https://www.tensorflow.org/
Thus, there is a need to understand how the model sees the input image by looking at the output of its intermediate layers. This provides the specifications about the working of these layers and how it has contributed to the classification output. Thus, in Fig.5 we have included the layer activation visualization of the pooling layer succeeding the Attention unit for a Malware as well as a Benign file sample, respectively.

As much as analysing the activation of the layer is significant for developing a deeper insight into our model, it is equally crucial to understand which parts of the image received more attention by the model. Heatmaps are used for this purpose where it uses color, the way a bar graph uses height and width for data visualization. A heatmap is an array like representation of scores corresponding to each pixel which indicates the relevance of each pixel for taking a classification decision. A heatmap is a subspace composed of pixels with high relevance [26]. Fig. 6(a) and 6(b) represent a sample malware and benign image feature representation using heatmaps.

Apart from visually understanding the functionality and behaviour of our model, we utilize a tool for predicting the probability of our classification decision using the Receiver Operating Characteristic (ROC) curve. It is a curve plotting the FPR on the x-axis as against the TPR on the y-axis for different threshold values between 0.0 and 1.0 for each class. This facilitates an understanding of the false malware prediction rate versus the true prediction rate using the ROC curve plot. This is an essential tool to evaluate our model especially for a task like Malware detection since withholding a benign file by wrongly assuming that it is a malware might pose a threat to availability, which is an integral part of an efficient security system. The ROC AUC score obtained was 0.9992.

The proposed method results (refer Table 3) have been compared with two existing CNN based architectures from the literature. The results have also been compared using conventional Machine Learning algorithms for classification viz. Support Vector Machine Classifier with Radial Basis Function (RBF) kernel and k-Nearest Neighbour. The features used for classification are the texture based GIST features. All the classifiers have been used with 5-fold cross validation. Detailed results for the proposed methodology are given in Table 4. Results show that we obtain an almost diagonal Confusion Matrix with a mis-classification of 2 malware files (False Negatives) and 7 benign files (False Positives). For the Support Vector Machine Classifier, a GridSearchCV module from the scikit-learn Machine learning library in Python was used to give the best possible hyperparameters among Linear and RBF kernels and C values ranging from 1 to 10, to test our proposed methodology with. The optimal classifier was with an RBF kernel with C value equal to 10.

B. Discussion and Limitations

The dataset collected for this work consists of different variants of Windows malware which were converted into images by the method used by [27]. This dataset consists of Malware and Benign image files with labels which was used for our experiments. In this paper, the proposed model demonstrated good accuracy for detecting malware over the standard existing methods using CNNs and texture analysis. However, the model has a limitation that it cannot take variable length inputs. This implies that either during bytes to image conversion all the files have to be created with equal lengths by padding, which might result in redundancy or on the contrary, the created images have to be cropped to a uniform size, which causes information loss. This might be a critical compromise, since the bytes lost during cropping might be crucial for classification. This challenge needs to be overcome to handle polymorphic malware and zero-day attacks. Thus, Spatial Pyramid networks can be explored, since they are useful for variable length inputs [29]. Another challenge is to verify the proposed architecture with unknown malware such as using adversarial methods. This task is essential to be able to generalize the proposed architecture over zero-day attacks.

C. Conclusion and Future Works

In this paper, we propose a Residual Attention based Malware detection system. Residual Attention is applied to enable

| TABLE 4: Detailed Metrics for Proposed Methodology. |
|-----------------|--------------|-------------|--------------|--------------|
| Accuracy        | Precision    | Recall      | F1-score     |
| 99.25%          | 99.25%       | 99.25%      | 99.25%       |
the model to acquire a global understanding of the images with the help of its Mask and Trunk branches for feature extraction and evaluation. We propose an architecture with a single Residual Attention unit and compare the results with different numbers of such units. We train our model on a binary class dataset with Malware and Benign images. The use of Residual Attention proves to improve the detection of malware as against existing CNN architectures and Machine Learning techniques using GIST features. We achieved 99.25% accuracy for our architecture which outperforms these traditional methods. The proposed method obtains an almost diagonal matrix with the False Negatives, which are primary to avoiding any intrusion, being almost nil. We interpreted our model behaviour by visualizing those regions of a malware which enabled the model to distinguish it from benign files, using Saliency maps and Heatmaps. To validate the achievement of such a high accuracy value by the proposed method, we used a t-sne plot which articulated the non-linear separability of the extracted features by our model. In the future, we will continue to test our model on larger datasets and for different types of malware files. We also wish to explore other Attention based mechanisms which could computationally be less expensive as compared to the Residual Attention.

REFERENCES

[1] Srinivasan, S., Ravi, V., Sowmya, V., Kirichen, M., Nouroeddine, D. B., Anivilla, S., & Kp, Soman. (2020, March). Deep convolutional neural network based image spam classification. In 2020 6th Conference on Data Science and Machine Learning Applications (CDMA) (pp. 112-117). IEEE.

[2] Kirichen, M., & Al'tooobaeva, R. (2019, November). Towards Optimizing the Placement of Security Testing Components for Internet of Things Architectures. In 2019 IEEE/ACS 16th International Conference on Systems and Applications (AICCSA) (pp. 1-2). IEEE.

[3] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. Pham, S. K. Padannayil and K. Simran, “A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities,” in IEEE Transactions on Industry Applications, vol. 56, no. 4, pp. 4436-4456, July-Aug. 2020, doi: 10.1109/TIA.2020.2971952.

[4] “Cyber Security Report 2020”, by National Technology Security Coalition, April 6, 2020.

[5] Young Han Choi, Byoung Jin Han, Byung Chul Bae, Hyung Geun Oh and Ki Wook Sohn, “Toward extracting malware features for classification using static and dynamic analysis,” 2012 8th International Conference on Computing and Networking Technology (INC, ICCIS and ICMIC), Guemongu, 2012, pp. 126-129.

[6] R. Vinayakumar & Kp, Soman & Alazab, Mamoun & Srinivasan, Srijam & Ketha, Simran. (2020). A Comprehensive Tutorial and Survey of Applications of Deep Learning for Cyber Security. 10.36227/techrxiv.11473377.v1

[7] Gandotra, Ekta & Bansal, Divya & Sofat, Sanjeev. (2014). Malware Analysis and Classification: A Survey. Journal of Information Security. 05. 56-64. 10.4236/jis.2014.52006.

[8] M. Chowdhury, A. Rahman and R Islam, Malware analysis and detection using data mining and machine learning classification. In: Abawajy J, Choo K-KR, Islam R (eds) International conference on applications and techniques in cyber security and intelligence: applications and techniques in cyber security and intelligence. Springer International Publishing, Cham, pp. 266–274, 2018.

[9] Jain, Mugdha & Andreopoulos, William & Stamp, Mark. (2020). Convolutional neural networks and extreme learning machines for malware classification. Journal of Computer Virology and Hacking Techniques. 10.1007/s11416-020-00354-y.

[10] Souri, Ali Rezae Hosseini, Rahil. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. 8. 1-22. 10.1186/s13673-018-0125-x.

[11] Kolter, Jeremy & Maloof, Marcus. (2006). Learning to Detect and Classify Malicious Executables in the Wild. Journal of Machine Learning Research. 6. 2721-2744.

[12] Nataraj, Lakshmanan Karthikeyan, Shanmugavadivel & Jacob, Grégoire & Manjunath, B.. (2011). Malware Images: Visualization and Automatic Classification. 10.1145/2016904.2016908.

[13] Han, KyoungSoo & Lim, Jae & Im, Eul Gyu. (2013). Malware analysis method using visualization of binary files. Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS 2013. 317-321. 10.1145/2513228.2513284.

[14] Nataraj, Ali Rezae Hosseini. (2018). A state-of-the-art survey of malware detection approaches using data mining techniques. 8. 1-22. 10.1186/s13673-018-0125-x.

[15] Kolter, Jeremy & Maloof, Marcus. (2006). Learning to Detect and Classify Malicious Executables in the Wild. Journal of Machine Learning Research. 6. 2721-2744.

[16] Nataraj, Lakshmanan & Yegneswaran, Vinod & Porras, Phillip & Zhang, Jian. (2011). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. 10.1109/2406684.2046689.

[17] Naeem, H., Ullah, F., Naeem, M. R., Khalid, S., Vasan, D., Jabbar, S., & Saeed, S. (2020). Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. Ad Hoc Networks, 102154.

[18] Yajamanam, Sravani & Selvin, Vineet & Di Troia, Fabio & Stamp, Mark. (2018). Deep Learning versus Gist Descriptors for Image-based Malware Classification. 553-561. 10.5220/006685805530561.

[19] Vasan, Danish & Alazab, Mamoun & Vasan, Sobia & Naeem, Hamad & Safaei, Babak & Zheng, Qian. (2020). IMCFN: Image-based Malware Classification using Fine-tuned Convolutional Neural Network Architectures. Computer Networks. 171. 107138. 10.1016/j.comnet.2020.107138.

[20] Karanja, Mwangi & Masupe, Sheden & Jeffrey, Manda. (2020). Transfer Learning for Internet of Things Malware Analysis. 10.1007/978-3-030-38501-9_20.

[21] Cui, Z., Du, L., Wang, P., Cai, X., & Zhang, W. (2019). Malicious code detection based on CNNs and multi-objective algorithm. Journal of Parallel and Distributed Computing. 129, 50-58.

[22] Xiao, Wentao & Zhang, Bin & Xiao, Xi & Kumar, Arun & Zhang, Weihe & Zhang, Jiajia. (2019). Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. Future Generation Computer Systems. 10.1016/j.future.2019.09.025.

[23] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[24] Min Wang and Baoyuan Liu and Hassan Foroosh. (2016). Design for Efficient Convolutional Layers using Single Intra-channel Convolutions, Topological Subdivision and Spatial “Bottleneck” Structure. ArXiv1608.04337.

[25] Wang, Fei & Jiang, Mengqing & Qian, Chen & Yang, Shuo & Li, Cheng & Zhang, Honggang & Wang, Xiaogang & Tang, Xiaoou. (2017). Residual Attention Network for Image Classification. 6450-6458. 10.1109/CVPR.2017.683.

[26] Srivastava, Rupesh & Greff, Klaus & Schmidhuber, Jürgen. (2015). Highway Networks.

[27] S. Choi, S. Jang, Y. Kim and J. Kim, Malware detection using malware image and deep learning, 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2017, pp. 1193-1195, doi: 10.1109/ICTC.2017.81090895.

[28] Samek, Wojciech & Binder, Alexander & Montavon, Gregoire & Laschikin, Sebastian & Müller, Klaus-Robert. (2017). Evaluating the Visualization of What a Deep Neural Network Has Learned. IEEE Transactions on Neural Networks and Learning Systems. 28. 2660-2673. 10.1109/TNNLS.2016.2599820.

[29] Srinivasan, Srijam & R, Vinayakumar & Vishvanathan, Sowmya & Alazab, Mamoun & Kp, Soman. (2020). Multi-scale Learning based Malware Variant Detection using Spatial Pyramid Pooling Network. 740-745. 10.1109/INFOCOMWKSHP580562.2020.9162661.