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 Vidyapeetham Coimbatore, India.
Email: shamikaganesan@gmail.com
†Center for Artificial Intelligence, Prince Mohammad Bin Fahd University, Khobar, Saudi Arabia.
Email: vinayakumarr77@gmail.com, vravi@pmu.edu.sa
‡Faculty of CSIT, Al-Baha University, Saudi Arabia. and ReDCAD Laboratory, University of Sfax, Tunisia.
Email: moez.krichen@redcad.org
§Department of Computer Science, College of Computers and Information Technology, Taif University, Saudi Arabia.
Email: r.robai@tu.edu.sa

***Abstract*—Recent advancements in Cyber Security has amalgamated the strengths of Artificial Intelligence and Human Intelligence for Intrusion Detection. The colossal increase in the volume of new malwares 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. However, to cater to the complexity of such a data-intensive environment generalizability of malware detection becomes the key to creating a successful anti-malware system. 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 using Convolutional Neural Networks(CNNs) have helped in capturing local features, Attention based mechanisms play a vital role in detecting polymorphic malware. Hence, in this paper, we explore the use of an attention based mechanism known as Residual Attention for malware detection and compare this with existing CNN based methods and conventional Machine Learning algorithms with the help of GIST features. The proposed method outperformed traditional malware detection methods which use Machine Learning and CNN based Deep Learning algorithms, by demonstrating an accuracy of 99.25%.**

Keywords: CyberCrime, Cyber Security, Residual attention, Convolutional Neural Network, Deep learning.

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 numbers 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 the 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 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.

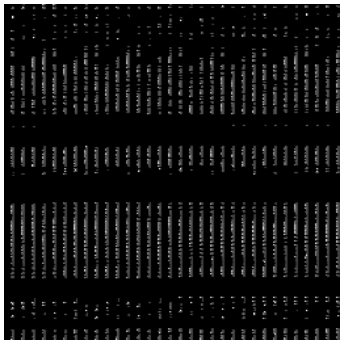

Fig. 1.a

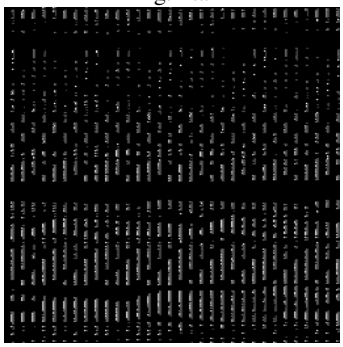

Fig. 1.b

Fig. 1: 1(a) and 1(b) represent two sample malware images from the dataset used for our experiments. The images show the variation in spatial information of different malware which cannot be efficiently generalized using only Convolutional layers.

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

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

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 generalizability in 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	Convolutional Neural Networks (CNN), Transfer Learning	Efficient classification. Requires generalizability to tackle polymorphic malware.
[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**	**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.**

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], [30], [31].

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 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, he 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.

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

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

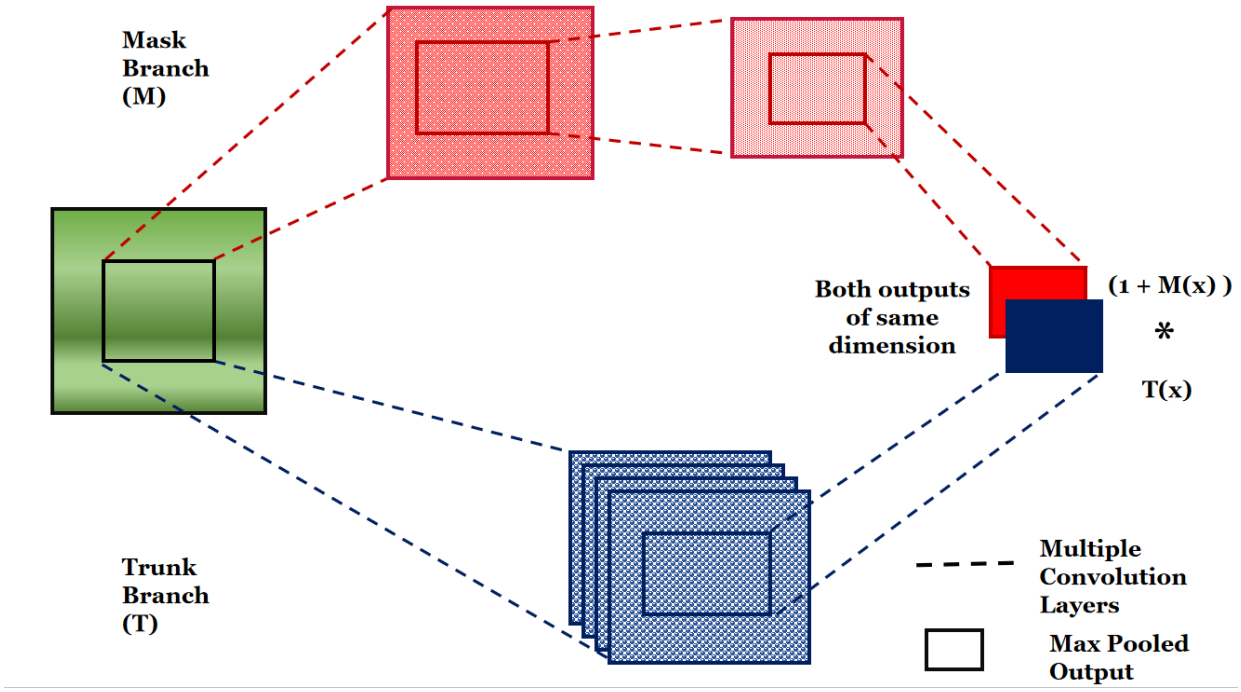

Fig. 2: Mask(M) and Trunk(T) branches of the Residual Attention unit for input (x).

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 architecture.

III. Background

A. Attention Networks

CNNs form the building blocks for most of the Computer Vision Deep Learning techniques. An image is just a multidimensional 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 more 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 weights that would be used to weigh the output features produced by the Trunk branch and also acts as gates for the Trunk branch during backpropagation to ensure that Gradient descent does not stall,

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

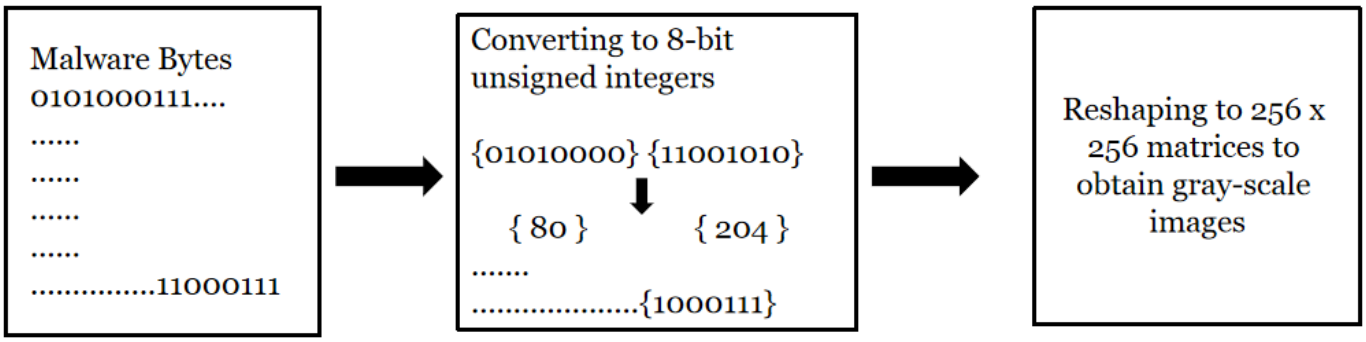

Fig. 3: Conversion of malware byte information into images

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.

Number of Residual Pre-activation Units	Number of Epochs	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
0	**100**	**99.25**	**99.25**	**99.25**	**99.25**
1	100	98.92	98.92	98.92	98.92
4	100	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]
Accuracy	**99.25**%	98.34%	97.55%	96.51%	95.60%

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 weightage 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 :

$$A_{j,k}(x) = M_{j,k}(x) * T_{j,k}(x) \tag{1}$$

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

$$\frac{\partial M(x,\alpha)T(x,\beta)}{\partial \beta} = M(x,\alpha)\frac{\partial T(x,\beta)}{\partial \beta} \tag{2}$$

where α are the parameters associated with the Mask branch and β 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 :

$$A_{j,k}(x) = (M_{j,k}(x) + 1) * F_{j,k}(x) \tag{3}$$

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).

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 (gray-scale). 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. Fig. 3 explains this in detail. 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

[1]https://scikit-learn.org/stable/

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

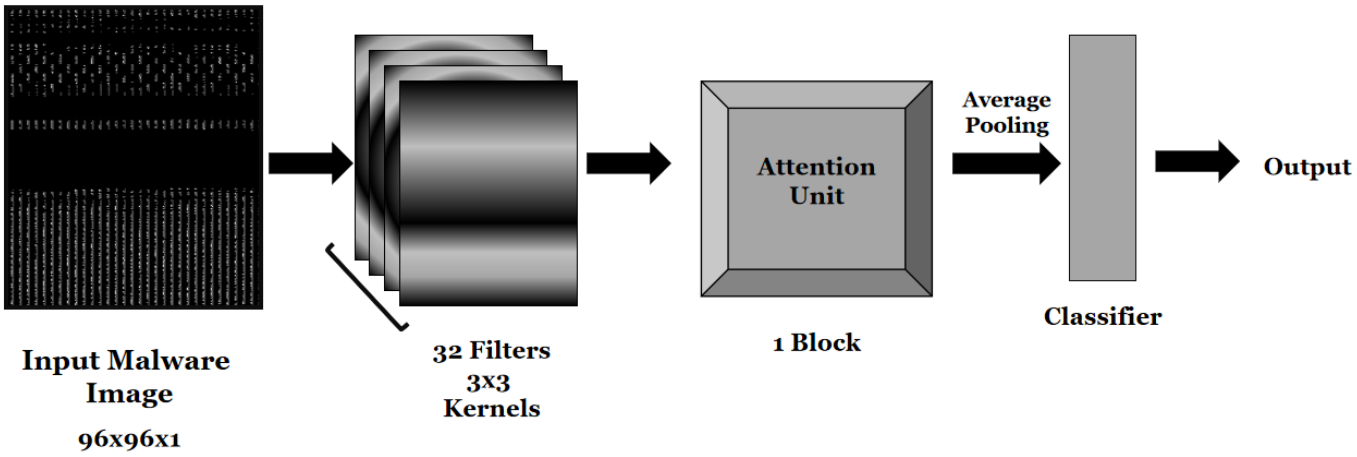

Fig. 4: Block Diagram for our proposed architecture. The input image is resized to 96x96 and fed into a 2D convolutional layer followed by Max-pooling. The output from the Attention unit undergoes Average Pooling and is fed to a fully connected layer network with dropouts after each fully-connected layer. The output is finally fed into the classifier to determine whether the image is a malware or benign image.

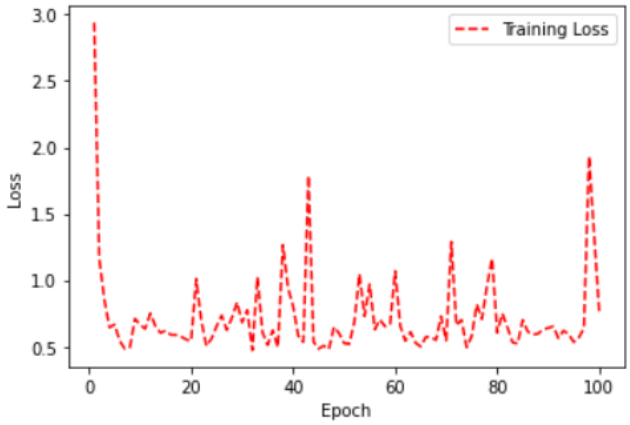

Fig. 5: Graph showing the Training loss for our proposed model against the number of epochs.

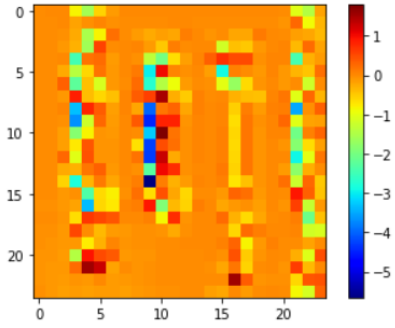

Fig. 6.a

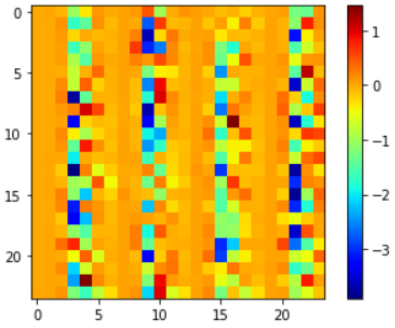

Fig. 6.b

Fig. 6: Layer Activation of one channel for Average Pooling after the Attention unit. 6(a) shows layer activation for a malware image. 6(b) shows layer activation for a benign image.

information of malware images and for increasing robustness for polymorphic malware. Our model consists of an input layer which takes in gray scale 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 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. 4. 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

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

epochs. Fig. 5 shows the graph for training loss plotted against the number of epochs of training.

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 $\longleftarrow image$
2dConvolution (32 filters, 3x3 kernel, 1x1 stride) $\longleftarrow Input$
MaxPool(2x2 size, 2x2 stride, 'same' padding) $\longleftarrow 2dConvolution$
Attention ([32, 64, 128] filters) $\longleftarrow MaxPool$
AvgPool (2x2 size, 2x2 stride) $\longleftarrow Attention$
Flatten () $\longleftarrow AvgPool$
Dense (n_class, activation) $\longleftarrow Flatten$
return Dense

V. Performance Evaluation

A. Statistical Measures

To measure the effectiveness of any model, the confusion matrix is analysed. Confusion Matrix is a table representing the performance of a Classifier with a class-wise split of the number of correctly predicted outcomes and the number of incorrectly predicted outcome. A confusion matrix for a malware classification is as follows :

	True Malware	True Benign
Predicted Malware	True Positive(TP), Malware	False Positive(FP), Benign
Predicted Benign	True Negative(TN), Benign	False Negative(FN), Malware

Using the above metrics, we have estimated the standard evaluation metrics, viz. Accuracy, Precision, Recall and F1-score. These are defined as follows :

Accuracy: It denotes the fraction of correct predictions (TP and TN) over total predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Precision: It denotes the fraction of correct positive predictions over the total number of all positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall: It denotes the fraction of correct positive predictions over the total number of all samples belonging to the specific. Recall is also known as True Positive Rate (TPR)

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1-score: F1-score denotes the harmonic mean between the recall and precision values.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

False Positive Rate (FPR) : FPR denotes how often a positive class is predicted when the actual outcome is negative.

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

ROC AUC curve : The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems which plots the probability curve of the TPR against FPR at various threshold values. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

B. 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 Keras[3] 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 a lesser 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.

The t-SNE plot for the penultimate layer for obtained results is shown in Fig.8. 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 maps represent 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

[2]https://colab.research.google.com/notebooks/intro.ipynb
[3]https://keras.io/
[4]https://www.tensorflow.org/

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

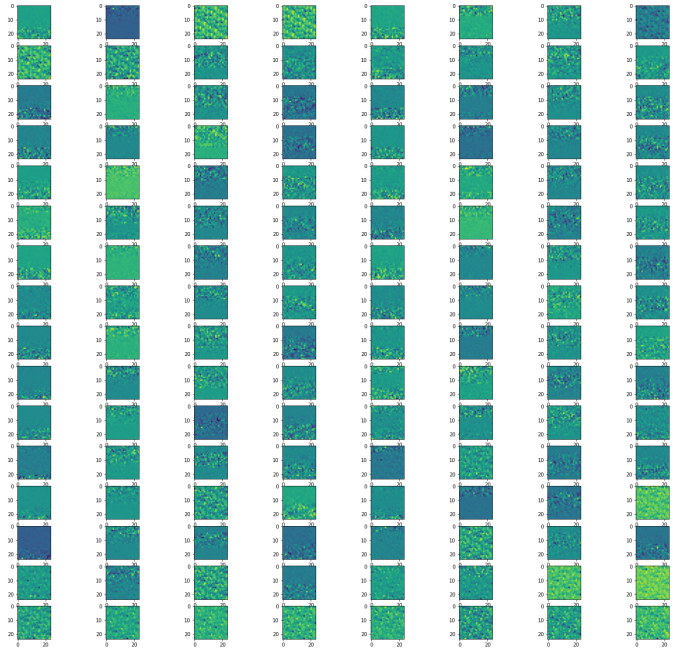

Fig. 7.a

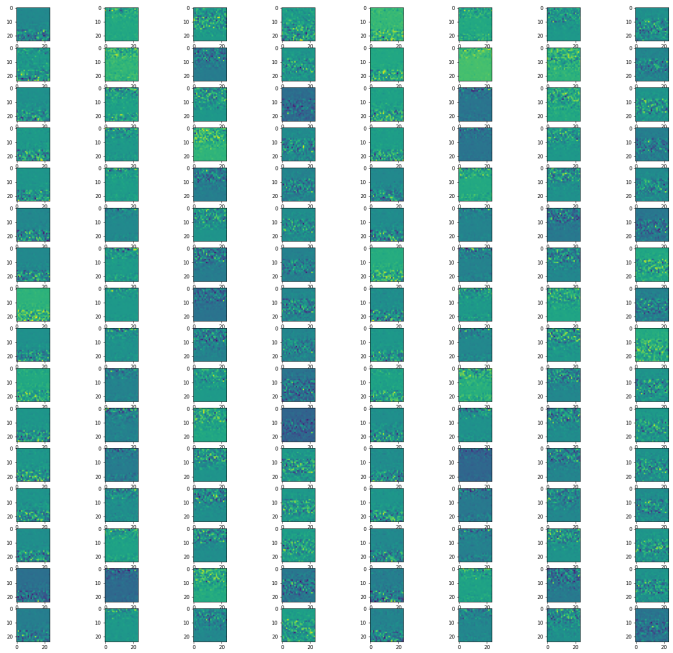

Fig. 7.b

Fig. 7: Layer Activation for Average Pooling after the Attention unit. 7(a) shows all channels for malware image 7(b) shows all channels for benign image.

maps for Malware and Benign images has been carried out, which is shown in Fig. 11(a) and Fig.11 (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.

$$\frac{\partial(PredictedClass)}{\partial(PixelValue)} \tag{9}$$

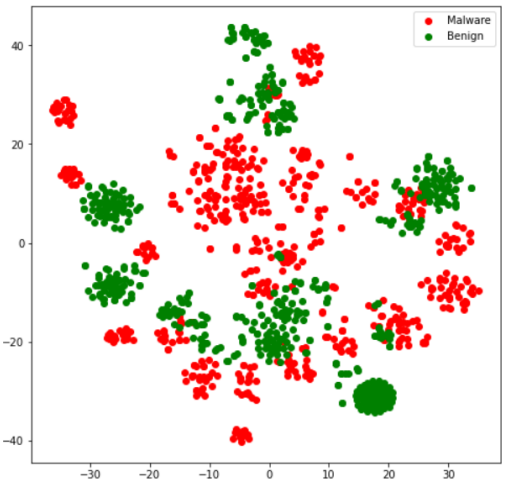

Fig. 8: t-SNE plot for the penultimate layer of the proposed architecture for testing images.

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. 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.6 and Fig.7 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. 10(a) and 10(b) represent a sample malware and benign image feature representation using heatmaps. Fig. 9 shows heatmaps for all the misclassified files by our proposed model.

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

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

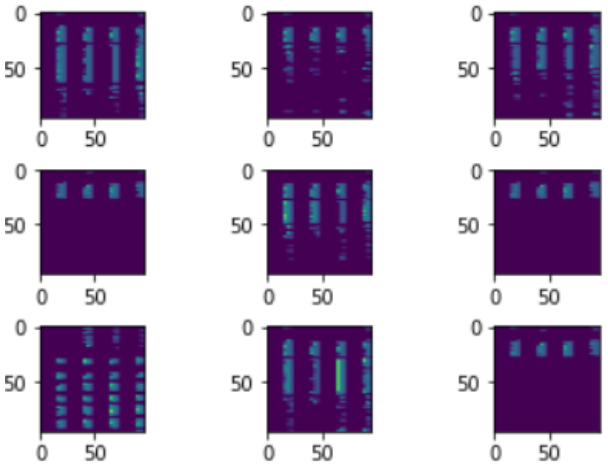

Fig. 9.a

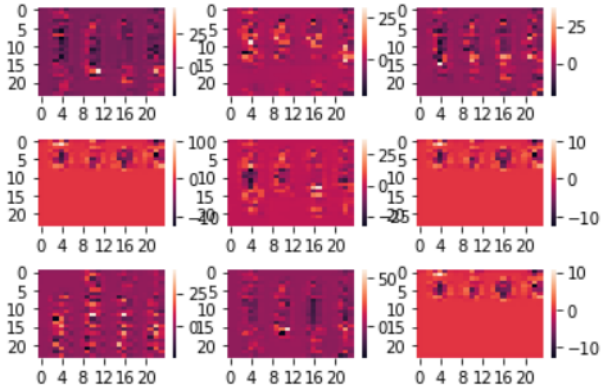

Fig. 9.b

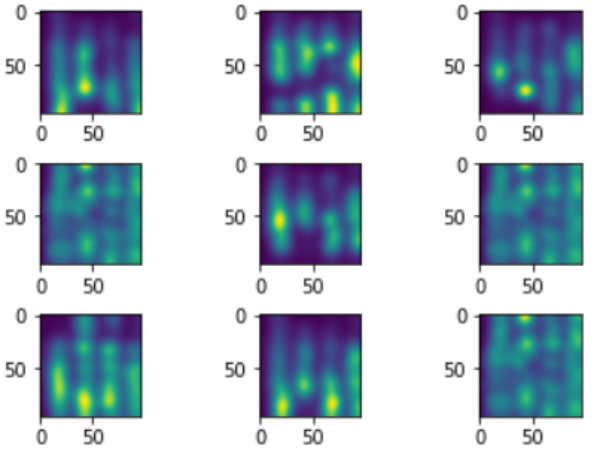

Fig. 9.c

Fig. 9: (a)Images of malware and benign files which are misclassified. 9(b) Heatmaps for the features extracted for the corresponding misclassified images. 9(c) Saliency maps for the misclassified images.

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. Fig. 12 shows the ROC curve for the proposed model. The ROC AUC score obtained was 0.9992.

The proposed method results (refer Table 3) have been compared with an existing CNN based architecture with the same number of Convolution layers as the proposed architecture along with a classification layer to classify the images. The proposed method 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 and 5. Results show that we obtain an almost diagonal Confusion Matrix. 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.

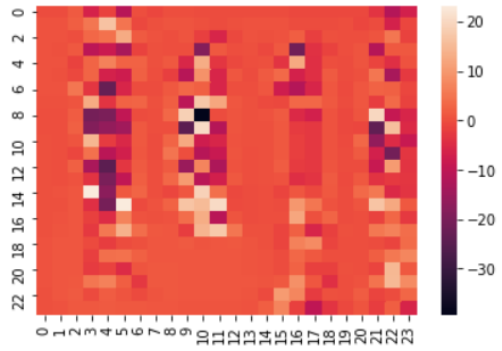

Fig. 10.a

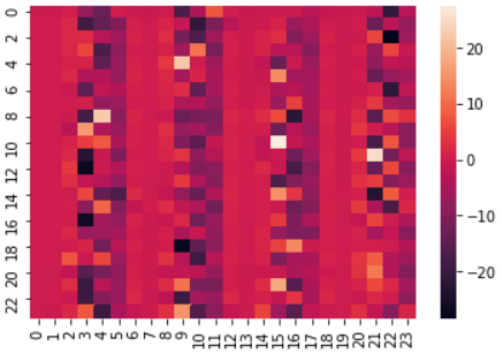

Fig. 10.b

Fig. 10: (a) Heatmap representation of the features extracted for a malware image. (b) Heatmap representation of the features extracted for a benign image.

C. 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

TABLE 4: Detailed Metrics for Proposed Methodology

Accuracy	Precision	Recall	F1-score
99.25%	99.25%	99.25%	99.25%

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 in computation or on the contrary, the created images have to be cropped to a uniform size, which might result in information loss. This might be a critical compromise, since the bytes lost during cropping might be the section of the malware learnt by the model to classify it correctly. This challenge needs to be overcome, especially for being robust against 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.

TABLE 5: Confusion Matrix for Proposed Methodology

	True Malware	True Benign
Predicted Malware	599	7
Predicted Benign	2	598

D. Generalization of the Proposed Work

Deep Learning is transforming into a prime tool in Artificial Intelligence applications, especially in Computer Vision. This kindled the use of time saving methods such as Transfer Learning using pre-trained models [18]. A pre-trained model is one which is trained on a benchmark dataset with different variations in data, which would have grasped the features that are necessary to solve a problem similar to the one we have in hand. Due to the computational cost of training such models, the trained weights of these models are readily available to be imported into our application (e.g. VGG, Inception, MobileNet) [18]. On similar lines, we tried to experiment with the proposed model for evaluating its performance on other malware datasets apart from Windows Malware. More specifically, we collected Android malware dataset and evaluated our model's performance. Our proposed method gave an accuracy of 33.7% for the Android malware. This is due to the variation in the structural information of Windows malware and Android malware. The obtained accuracy shows that using various levels of tuning, our model could show an improved performance for a transfer learning based approach.

In addition, we also observe that the datasets we have used for training and testing so far are balanced datasets. Using an imbalanced dataset could give lower results, which can be mitigated using penalized models which impose additional weights to the class with lesser samples.

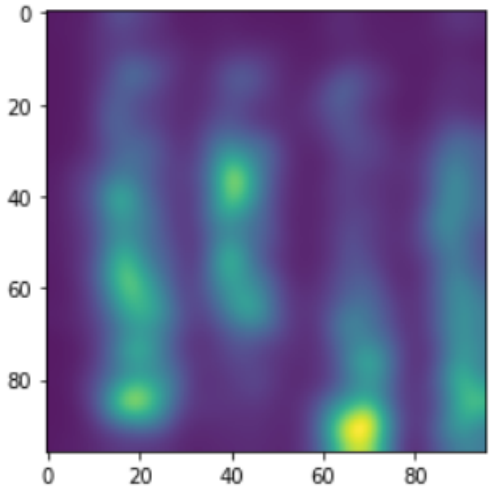

Fig. 11.a

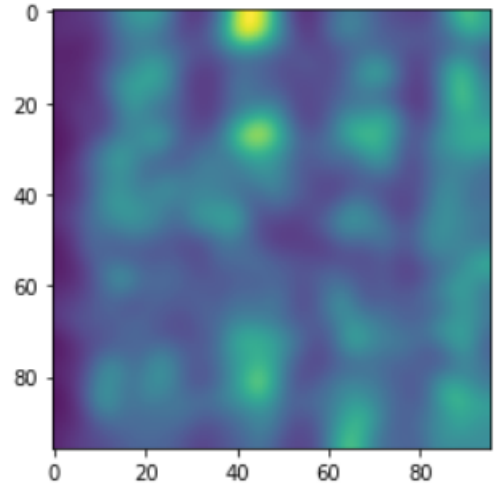

Fig. 11.b

Fig. 11: (a) Saliency map for Benign image.(b) Saliency map for Malware image.

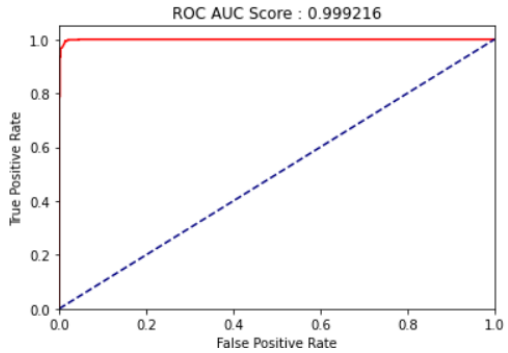

Fig. 12: ROC curve for Proposed Method

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

E. Conclusion and Future Works

In this paper, we propose a Residual Attention based Malware detection system architecture using a visual representation of Malware byte information. Residual Attention is applied to enable the model to acquire a global understanding of the images with the help of its Mask and Trunk branches for feature extraction and evaluation, unlike a completely CNN based architecture which focuses only on local features. 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 images for Malware and Benign byte files. The use of Residual Attention proves to improve the detection of malware as against conventional CNN based architectures and Machine Learning based classification techniques using texture based GIST features. We have compared our results using the same. We achieved 99.25% accuracy for our architecture which improves the accuracy over the traditional methods. As shown in the confusion matrix, Residual Attention obtains an almost diagonal matrix. We also find that the False Negatives, which are primary to avoiding any Intrusion, are almost nil. 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., Krichen, M., Noureddine, 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] Krichen, M., & Alroobaea, R. (2019, November). Towards Optimizing the Placement of Security Testing Components for Internet of Things Architectures. In 2019 IEEE/ACS 16th International Conference on Computer 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), Gueongju, 2012, pp. 126-129.

[6] R, Vinayakumar & Kp, Soman & Alazab, Mamoun & Srinivasan, Sriram & 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] Sun, Guosong & Qian, Quan. (2018). Deep Learning and Visualization for Identifying Malware Families. IEEE Transactions on Dependable and Secure Computing. PP. 1-1. 10.1109/TDSC.2018.2884928.

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

[12] 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. 10.1145/2513228.2513294.

[13] Souri, Alireza & 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.

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

[15] Nataraj, Lakshmanan & Yegneswaran, Vinod & Porras, Phillip & Zhang, Jian. (2011). A comparative assessment of malware classification using binary texture analysis and dynamic analysis. 10.1145/2046684.2046689.

[16] 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.

[17] Yajamanam, Sravani & Selvin, Vikash & Di Troia, Fabio & Stamp, Mark. (2018). Deep Learning versus Gist Descriptors for Image-based Malware Classification. 553-561. 10.5220/0006685805530561.

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

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

[20] 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.

[21] R, Vinayakumar & Alazab, Mamoun & Kp, Soman & Poornachandran, Prabaharan & Venkatraman, Sitalakshmi. (2019). Robust Intelligent Malware Detection Using Deep Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2906934.

[22] Xiao, Wentao & Zhang, Bin & Xiao, Xi & Kumar, Arun & Zhang, Weizhe & 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 of Efficient Convolutional Layers using Single Intra-channel Convolution, Topological Subdivisioning 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.2017.8190895.

[28] Samek, Wojciech & Binder, Alexander & Montavon, Gregoire & Lapuschkin, 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, Sriram & R, Vinayakumar & Vishvanathan, Sowmya & Alazab, Mamoun & Kp, Soman. (2020). Multi-scale Learning based

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).

Malware Variant Detection using Spatial Pyramid Pooling Network. 740-745. 10.1109/INFOCOMWKSHPS50562.2020.9162661.

[30] Sriram, S., Vinayakumar, R., Alazab, M., Soman, K. P. (2020, July). Network Flow based IoT Botnet Attack Detection using Deep Learning. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 189-194). IEEE.

[31] Vinayakumar, R., Soman, K. P., Poornachandran, P., Alazab, M., Jolfaei, A. (2019). DBD: deep learning DGA-based botnet detection. In Deep Learning Applications for Cyber Security (pp. 127-149). Springer, Cham.

This is a preprint and accepted in 39th International Conference on Consumer Electronics, IEEE ICCE 2021 (http://www.icce.org/).