International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT2173117

Malware Threat Detection using Deep Neural Networks

Sriram Muralidharan
Department of Computer Science and Engineering, SSN College of Engineering, Kanchipuram, Tamil Nadu, India

ABSTRACT

Article Info
Volume 7, Issue 3
Page Number: 515-522

Publication Issue :
May-June-2021

Article History
Accepted : 03 June 2021
Published : 08 June 2021

Malware threat detection is one of the most challenging tasks in the field of Information Security and the shortage of qualified personnel makes it even harder for people to keep their information secure. Moreover, malware design has evolved continuously, making it even more difficult for people to protect themselves from malware attacks. Thus, it is the need of the hour to improve the existing malware threat detection systems with modern deep learning algorithms. This paper focuses on bringing together a comprehensive study of various deep learning solutions for the detection of malware from its PE file (Portable Executable File) byte streams.

Keywords: Static Malware Analysis, Threat Detection, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks and Autoencoders.

I. INTRODUCTION

Malware, or “malicious software,” is an umbrella term that describes any malicious program or code that is harmful to the computer system. Being hostile and intrusive, malware seeks to invade, damage, or disable computer systems, networks, tablets, and mobile devices, often by taking control over a device’s operations.

New malware comes out every four seconds. This statistic, from an industry report [1], illuminates just how constant the flow of new malware is. For every stain of malware that’s defeated, there’s a new one to take its place. Malware development has seen diversity in terms of architecture and features. This advancement in the competencies of malware poses a severe threat and opens new research dimensions in malware detection [2] [3].

A lot of malware these days are based on the Windows PE file format. The PE (Portable executable) file format commonly known as the executable file format (.exe) is used in both x86 and x64 architectures of the Windows Operating System. The PE file format is a data structure that contains the information necessary for the Windows OS loader to manage the wrapped executable code. This PE file in hindsight is a byte stream that can be visualized as an image [4] and can be subjected to deep learning algorithms for pattern recognition, feature extraction and so on. This paper focuses on giving a systematic study on the various algorithmic strategies that can be employed for malware threat detection and thereby attempting to bring about an effective strategy for the same.

The organization of this paper is as follows. In Section 2 (**Methodology**), An In-depth study of various algorithmic strategies used for malware detection will be explained. In Section 3 (**Results**), a comparison study of the various algorithmic strategies along with the necessary results obtained for each strategy will be presented. Discussed in Section 4 (**Conclusion**) a conclusion followed by the scope for improvement.

II. METHODOLOGY

A. Dataset Description

The dataset [5] consists of 51959 instances of raw PE byte streams that are rescaled to 32x32 greyscale images using the Nearest Neighbour Interpolation algorithm and then flattened to a 1024 bytes vector as shown in Fig. 1. PE malware examples were downloaded from virusshare.com. The PE files for "non-malware" examples were downloaded from portableapps.com and from Windows 7 x86 directories. These images were normalized and resampled before the further proceedings.

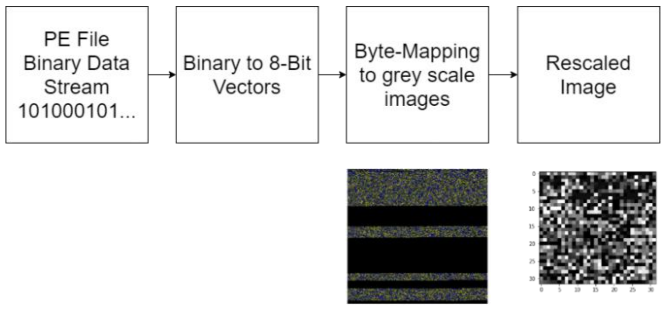

PE File Binary Data Stream 101000101...
Binary to 8-Bit Vectors
Byte-Mapping to grey scale images
Rescaled Image

Figure 1. Steps involved in visualizing malware as an image

B. Structuring Deep Neural Network Architectures

This paper focusses on incorporating the following architectures:

- Convolutional Neural Networks based classifier.
- Recurrent Neural Networks based classifier.
- Autoencoder for feature extraction followed by a Deep Neural Network classifier.

C. Construction of the CNN classifier

The intuition behind using a CNN classifier [6] is to leverage the ability of the model to recognize patterns in images that are invisible to the naked eye. The CNN model in this study follows the architecture as shown in Fig. 2., which comprises a series of convolutional and maxpooling layers followed by an output layer (dense layer) with the sigmoid activation. The output of the final layer is then subjected to a threshold function Θ(x) with Θ = 0.3 to determine if the image is a malware.

$$\Theta(x) = \begin{cases} 0, & x < \Theta \\ 1, & otherwise \end{cases}$$

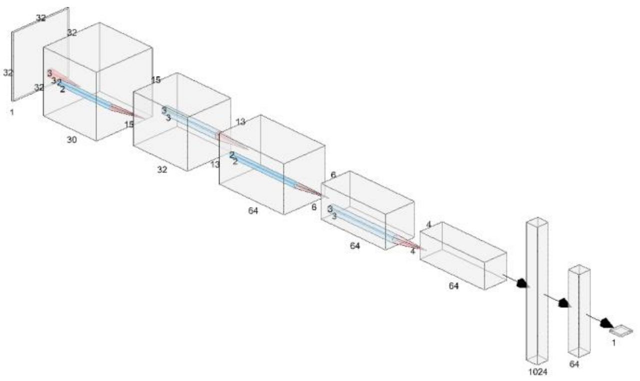
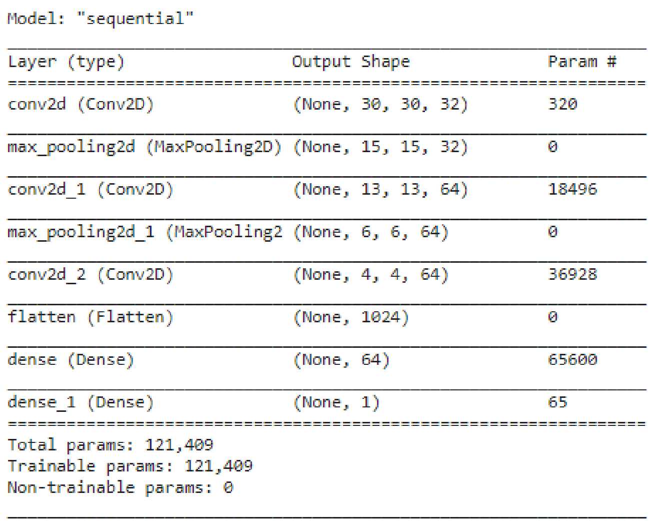


Figure 2. Proposed Architecture of the CNN Classifier



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 64)                65600
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 121,409
Trainable params: 121,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. Summary of the CNN Classifier model

D. Construction of RNN classifier

The intuition behind using an RNN model is that it is a network that works on the present input by taking into consideration the previous output (feedback) and storing in its memory for a short period of time (short-term memory) thus in hindsight it should be able to learn recurring patterns in the malware image. The recurring pattern can be attributed the obfuscation of code, which is a common pattern in malware. This RNN model uses LSTM (Long Short-Term Memory) [7] networks which are an extension of recurrent neural networks (RNNs) introduced to handle the vanishing gradient problem. Fig. 4. shows a single LSTM cell.

The detailed architecture of the RNN model can be seen in Fig 5. And similar to the CNN model the output of the RNN classifier is also sent to the threshold function Θ(x) with Θ = 0.3. Fig. 5 shows the complete architecture of the RNN model.

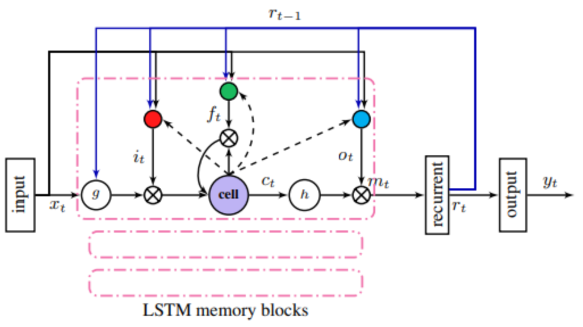


Figure 4. Structure of a single LSTM cell

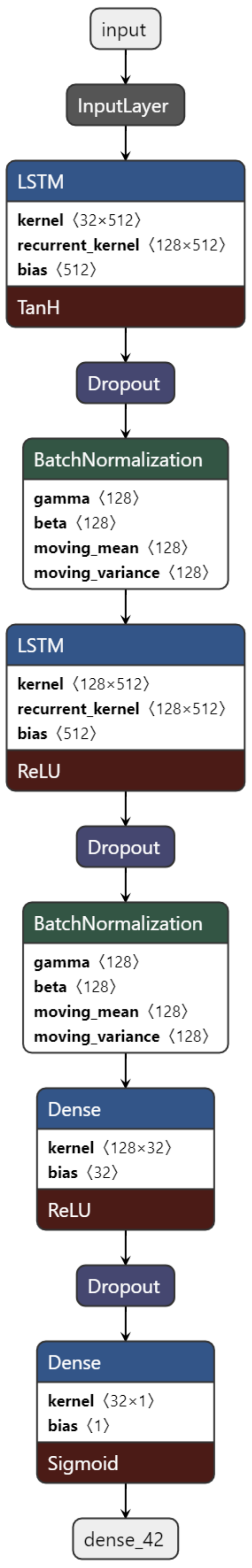


Figure 5. Architecture of the RNN model

D. Construction of the Autoencoder

Autoencoders [8] are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion. The architecture of the autoencoder is shown in Fig. 6. Autoencoders are primarily used for dimensionality reduction and feature extraction. In the proposed implementation we use a convolutional autoencoder [9]. The Convolutional Autoencoder consists of an Encoder circuit and Decoder circuit. The input for the encoder is the PE malware image which is then subjected to series of Convolutional layers and Maxpooling layers until a bottleneck layer with dimensions 32x1 is obtained as shown in Fig 7. This is the feature vector that is to be sent for classification. The decoder layer reconstructs the original input using transpose convolution while minimizing the loss, it's architecture is shown in Fig. 8.

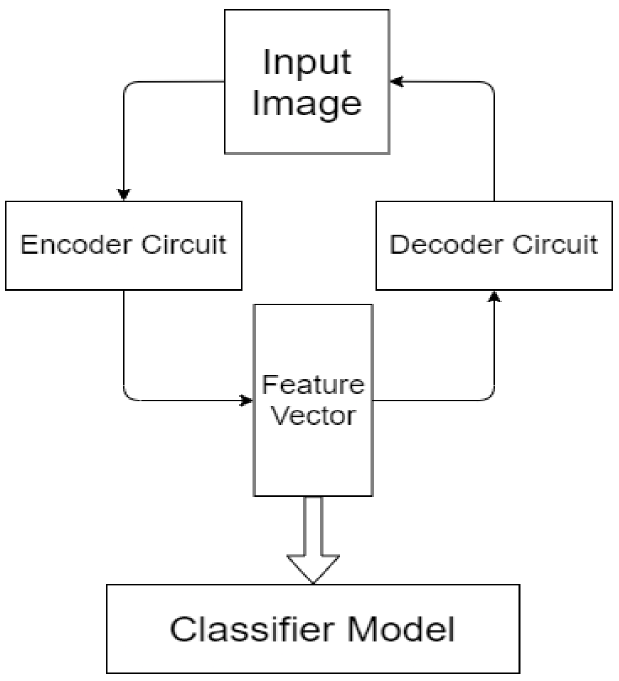
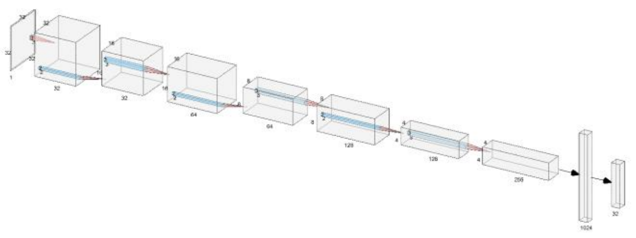
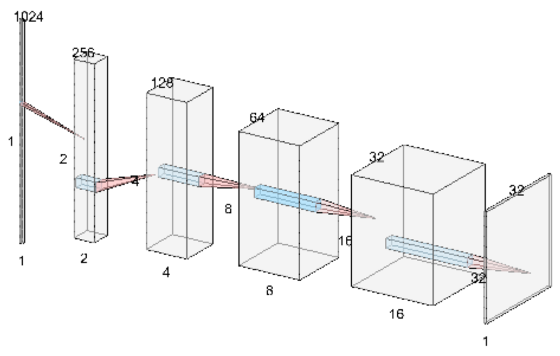


Figure 6. Architecture of the Autoencoder Model



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                32800
=================================================================
Total params: 420,640
Trainable params: 420,640
Non-trainable params: 0
```

Figure 7. Architecture of the Encoder Circuit



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 1024)              33792
_________________________________________________________________
reshape (Reshape)            (None, 2, 2, 256)         0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 4, 4, 128)         295040
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 8, 8, 64)          73792
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 16, 16, 32)        18464
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 32, 32, 1)         289
=================================================================
Total params: 421,377
Trainable params: 421,377
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. Architecture of the Decoder Circuit

Fig 9. shows how the autoencoder reconstructs the image. Thus, with the help of the autoencoder we have obtained the feature vectors of all the instances in the dataset. This collection of feature vector is sent for classification.

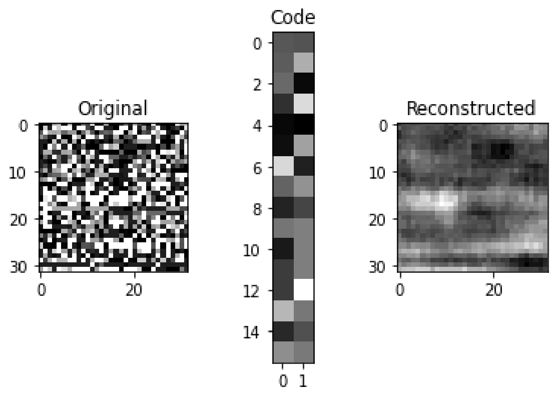


Figure 9. Reconstruction of Original Image

F. ANN Classifier

From various comparative studies [10] there are clear evidences that elucidates the better performance of a custom ANN network for classification over traditional ML techniques. Thus, a simple ANN model was built with the architecture shown in Fig. 10

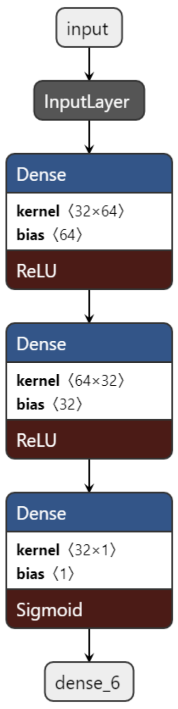


Figure 10. Architecture of the ANN classifier

III. RESULTS AND DISCUSSION

The models in the above section are tested on the basis of the following metrics [11]:

A. Confusion Matrix

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one, the basic structure of the confusion matrix can be seen in Fig 11. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The confusion matrices for various models can be visualized in Fig 12.

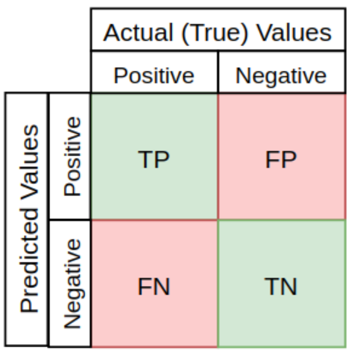
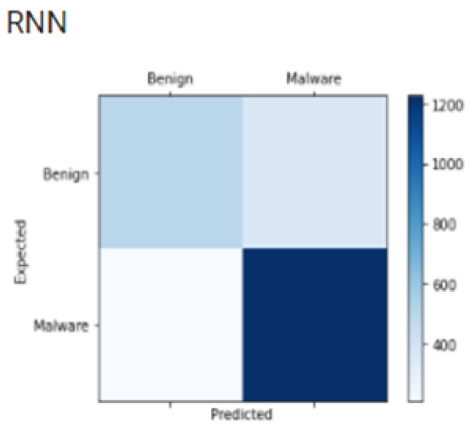


Figure 11. Structure of a confusion matrix



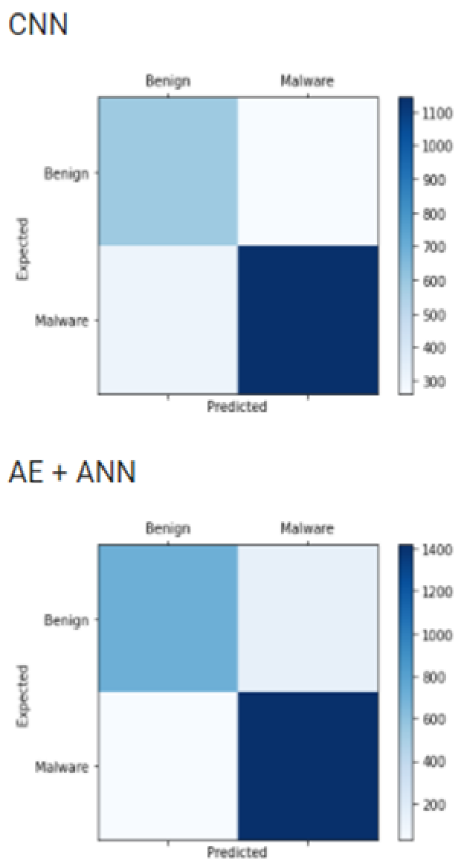


Figure 12. Confusion matrices for corresponding algorithms

B. Accuracy

Binary Accuracy calculates the percentage of predicted values that match with actual values for binary labels.

We calculate *Binary Accuracy* by dividing the number of accurately predicted records by the total number of records. The binary accuracy of the model are shown in table 1.

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

TABLE I. BINARY ACCURACY

S.no	Model	Binary Accuracy
1.	RNN Classifier	0.79503
2.	CNN Classifier	0.91467
3.	Autoencoder with ANN Classifier	0.92337

C. Precision, Recall and F1 score

Precision is a measure of how many of the positive predictions made are correct (true positives). The formula for it is:

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

Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data. It is sometimes also referred to as Sensitivity. The formula for it is:

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

F1-Score is a measure combining both precision and recall. It is generally described as the harmonic mean of the two given by the formula:

$$F1\ Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

TABLE III. PRECISION, RECALL AND F1 SCORE

S No.	Model	Evaluation Metric		
		Precision	Recall	F1-Score
1	RNN Classifier	0.8587	0.8346	0.8464
2	CNN Classifier	0.9191	0.9476	0.9331
3	Autoencoder + Classifier	0.9029	0.9836	0.9416

IV.CONCLUSION

From the studies conducted above we can clearly see that the Deep Learning solutions have proven to be very effective in Malware threat detection. Convolutional Networks and Autoencoders have captured the features of malware and Deep Neural

Networks have proven to be very effective classifiers. Viewing malware PE files as images has certainly opened a lot of scope of research and development of malware threat analysis in the field of image processing and deep learning. Thus, it is the time that the industry shifts towards finding Deep Learning based Solutions for solving the ever-prevailing threats of malware attacks. The models can be further improved with the help of transfer learning and by constructing more complex architectures. Thus, we can conclude that Deep Learning Based solutions can serve as an effective first layer classification for Malware Threat Detection.

V. REFERENCES

[1]. Ionut Ilascu. 2014. Four New Malware Samples Emerge Every Second. https://news.softpedia.com/news/Four-New-Malware-Samples-Emerge-Every-Second-457576.shtml

[2]. Satya Narayan Tripathy, Sisira Kumar Kapat, M. Soujanya, Susanta Kumar Das, "Static Malware Analysis : A Case Study", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 2, Issue 7, pp.135-141, September-2017.

[3]. Soumen Chakraborty, "Malware attack and Malware Analysis : A Research", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 5, Issue 3, pp.268-272, May-June-2019.

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

[5]. Angelo Oliveira, 2019. Malware Analysis Datasets: Raw PE as Image. (Nov 2019) Available at: https://dx.doi.org/10.21227/8brp-j220.

[6]. O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[7]. Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[8]. Bank, Dor & Koenigstein, Noam & Giryes, Raja. (2020). Autoencoders. ArXiv e-prints.

[9]. Agarwal, Akarsh & Motwani, Akash. (2016). An overview of Convolutional and AutoEncoder Deep Learning Algorithm.

[10]. M. Mishra and M. Srivastava, "A view of Artificial Neural Network," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), 2014, pp. 1-3, doi: 10.1109/ICAETR.2014.7012785..

[11]. Bhodia, Niket & Prajapati, Pratikkumar & Di Troia, Fabio & Stamp, Mark. (2019). Transfer Learning for Image-Based Malware Classification. 10.5220/0007701407190726.

[12]. Hossin, Mohammad & M.N, Sulaiman. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. International Journal of Data Mining & Knowledge Management Process. 5. 01-11. 10.5121/ijdkp.2015.5201.

[13]. Stamp, Mark & Alazab, Mamoun & Shalaginov, Andrii. (2021). Malware Analysis Using Artificial Intelligence and Deep Learning. 10.1007/978-3-030-62582-5.

Cite this article as :

Sriram Muralidharan, "Malware Threat Detection using Deep Neural Networks", International Journal of Scientific Research in Computer Science,

Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7, Issue 3, pp.515-522, May-June-2021. Available at doi : https://doi.org/10.32628/CSEIT2173117
Journal URL : https://ijsrcseit.com/CSEIT2173117