Visual Attention for Malware Classification

Naseem Alsadi*[a], Waleed Hilal[a], Onur Surucu[a], Alessandro Giuliano[a], Stephen A. Gadsden[a], John Yawney[b]
[a]McMaster University, 1280 Main St W, Hamilton, ON L8S 4L8; [b]Adastra Corporation, 8500 Leslie St #600, Thornhill, ON L3T 7M8

ABSTRACT

Amidst the extensive global integration of computer systems and augmented connectivity, there have been numerous difficulties within ensuring confidentiality, integrity and availability across all systems. Malware is an ever-present and persistent challenge for security systems of all sorts. Numerous malware detection methods have been proposed, with traditional approaches no longer providing the necessary protection against evolving attack methodologies and strategies. In recent years, machine learning for malware detection has been investigated with great success. In addition, the analysis of application operation code, or opcode, due to its unavoidable nature, can reveal necessary information about software intention. Visualization of opcode data allows for simple data augmentation and texture analysis. The proposed approach utilizes a simple visual attention module to perform a binary classification task on program data, focusing on visualized application opcode data. The proposed model is tested with an ARM-based Internet of Things (IoT) application opcode dataset. In addition, a comparative analysis, using numerous metrics, is conducted on the proposed model's performance along with several other algorithms. The results indicate that the proposed method outperformed all other tested techniques in accuracy, recall, precision, and F-score.

Keywords: Malware Detection, Artificial Intelligence, Deep Learning

1. INTRODUCTION

While computer systems have been adopted across numerous fields of application and have facilitated high-frequency communication and augmented system interconnectivity, the presence of malicious attacks has grown rapidly. Continuous development in the field of attack detection, specifically malware detection, has improved tremendously over the last decade [1]. However, attack complexity has also evolved, and traditional detection methods do not provide the necessary protection [2].

For instance, polymorphic malware, which is a form of malware that can mutate while preserving the fundamental structure and malicious intention, has become more prominent and the difficulty of detection has increased substantially [3]. Novel methods of detection have been explored by various researchers, with an exponentially growing body of literature dedicated to the employment of machine learning in the field of malware detection.

Machine learning as a field has grown tremendously, with an $8 billion market value in 2017 and an estimated market value of $117 billion by 2027 [4], [5]. The integration of machine learning across numerous domains and facets of daily life continues, and its incorporation within cybersecurity is no different. Figure 1 showcases the large increase in academic publications during the years 2013 and 2021 concerning machine learning and malware detection, generated using Google Scholar and IEEE.

Machine learning techniques have been widely employed across numerous applications in the field of malware detection. Most commonly, supervised learning techniques are employed to learn from a dataset of registered malware samples and predict on novel application data [6]. Numerous researchers have employed simple and complex machine learning techniques to detect malicious behaviour in a variety of environments, with the Internet of Things (IoT) and Android ecosystems becoming increasingly more popular [5–10].

Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV, edited by Tien Pham, Latasha Solomon, Proc. of SPIE Vol. 12113, 121131X
 · 0277-786X · doi: 10.1117/12.2619009

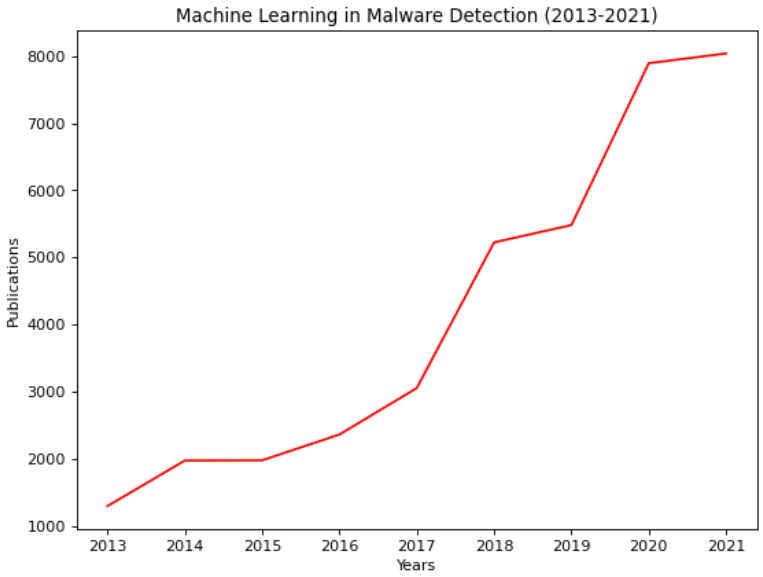

Figure 1: Publications on Machine Learning and Malware Detection (2013-2021)

In this paper, we postulate that with the aid of encoding, normalization, visualization and a visual attention network, malicious attacks can be efficiently detected. We employ a simple visual attention module, comprised of a squeeze and excitation network, convolutional neural network and long-short term memory. Visualised program data is sliced into glimpses and passed into the model, which performs a binary classification task, indicating if the input is representative of malicious intent. We test the proposed approach on an ARM-based IoT opcode dataset and report the performance against numerous other algorithms.

The performance of the proposed method is assessed using several metrics, namely accuracy, precision, recall and F-score:

- **True Positive (TP):** If a malicious sample is predicted as malware.
- **True Negative (TN):** If a malicious sample is predicted as goodware.
- **False Positive (FP):** If a goodware sample is predicted as malicious.
- **False Negative (FN):** If a malicious sample is predicted as goodware.

Accuracy: Indicates how well the proposed model can accurately predict malware and goodware.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (1)$$

Precision: Precision is the number of malicious samples that are correctly predicted, divided by the total number of samples that are predicted.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall: Recall is the number of malicious samples that are correctly predicted, over the total number of malicious samples in the testing set.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F-Score (F_1): F-Score is the harmonic mean of the precision and recall:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

2. METHODOLOGY

The proposed method relies heavily on the employment of visual attention to classify visualized opcode data. The cumulative proposed method involves three phases, namely the data encoding, visualization and the visual attention model.

Initially, the model is presented with a set of sequential opcodes from a specific program. The first step in the architecture is to encode the opcode data into a numerical representation, which can be completed with numerous techniques. In this paper, a dictionary of available opcodes is built apriori and the index value is utilized to represent each opcode. The encoding phase is completed when all the opcode sequences have been numerically represented.

Subsequently, each input sequence is restructured into a $H \times W \times G$ representation, where H is the height, W is the width and G is the number of glimpses. In addition, we utilize normalization to accelerate the convergence [13].

The visualization process allows for the analysis of various relationships within the input sequence, and the sequential nature of the data is preserved with the utilization of glimpses rather than whole images. This is explored in the results section below, where a convolutional neural network is utilized as a baseline model for comparison without the employment of glimpses.

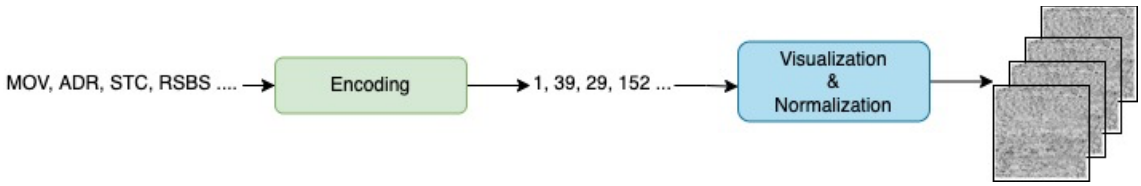

Figure 2: Opcode Encoding and Visualization

After completing the encoding and visualization phase of the proposed approach, the training phase is initiated. The fundamental components of the visual attention architecture are a squeeze and excitation network, a convolutional neural network and long-short term memory.

Squeeze and excitation networks are typically employed to produce an adaptive weighting for channel-wise data. However, in the proposed approach, the squeeze and excitation network is employed for the adaptive weighting of the glimpses taken from the original visualized image. This provides a simple glimpse attention mechanism that can be adaptively modified during the training phase to yield optimal performance.

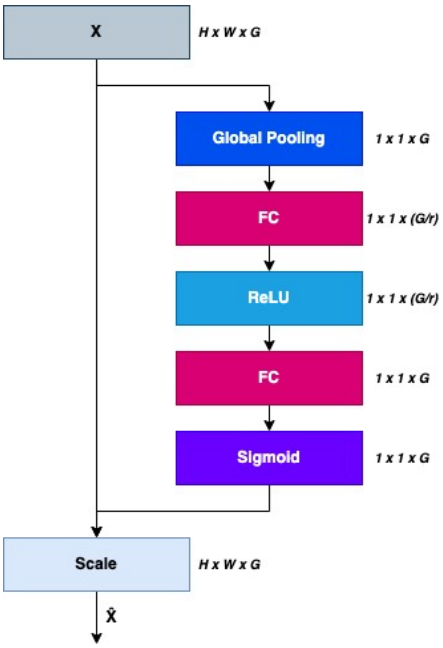

Figure 3: Squeeze and Excitation Network

The squeeze and excitation network will utilize a skip connection and a series of computations on the input along a variant path. The input of the network, which has a dimensionality of $H \times W \times G$, will be squeezed into a dimensionality of $1 \times 1 \times G$. This procedure is conducted with the use of a global average pooling operation, formulated as [14]:

$$z_G = F_{sq}(X_G) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_G(i,j) \tag{5}$$

Subsequently, two fully connected layers are employed to limit model complexity while augmenting generalization. The first fully connected layer will be followed by a piecewise linear activation and the second fully connected layer will be followed by a sigmoid activation.

$$s = F_{ex}(z, W) = \sigma\big(g(z, W)\big) = \sigma\big(W_2 \delta(W_1 z)\big), \tag{6}$$

where r is the reduction ratio controlling computational cost and capacity, σ and δ refers to the sigmoid and rectified linear unit function, $W_1 \in R^{\frac{G}{r} \times G}$ and $W_2 \in R^{G \times \frac{G}{r}}$.

The original sequence of glimpses is scaled by multiplying the channel scalar, s_G and feature map u_G

$$\hat{X} = F_{scale}(u_G, s_G) = s_G \cdot u_G, \tag{7}$$

where $\hat{X} = [\hat{x}_1, \hat{x}_2 \dots, \hat{x}_G]$ and $u_G \in R^{H \times W}$.

Subsequently, each glimpse will be analyzed with a convolutional neural network unit, however, we choose to implement a sequential visual variation using long-short term memory. This allows for greater modelling of the relationships between glimpses.

A convolutional neural network is a form of artificial neural network which is typically applied to visual data. The goal of the network is to learn spatial hierarchies of features [15]. With respect to the task at hand, the employment of the convolutional neural network will aid will the establishment of opcode relationships, with an emphasis on the spatial relationships between non-adjoining instructions.

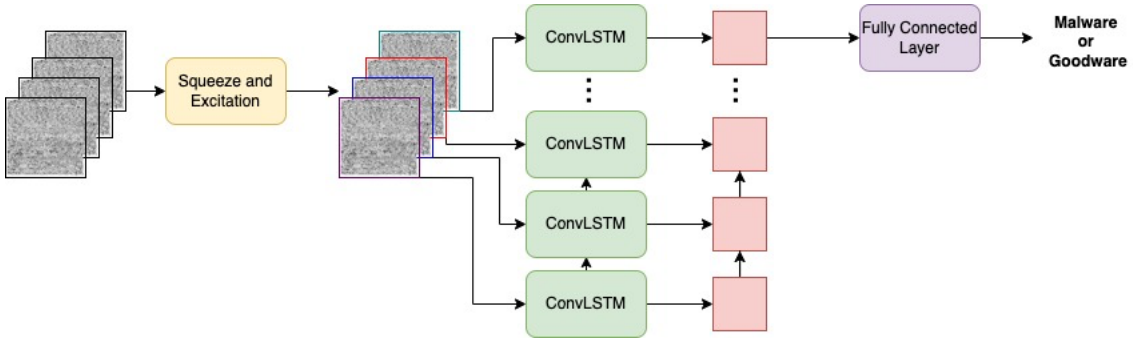

Figure 4: Model Architecture

Long short-term memory is a recurrent neural network architecture utilized for sequential data analysis. Where a sequential input is passed into the model and a classification or prediction is generated based on the sequence. The LSTM can be formulated as [16]:

$$i_t = \sigma(W_{xi}\, x_t + W_{hi} h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \tag{8}$$

$$f_t = \sigma\big(W_{xf}\, x_t + W_{hf} h_{t-1} + W_{cf} \circ c_{t-1} + b_f\big) \tag{9}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c) \tag{10}$$

$$o_t = \sigma(W_{xo}\, x_t + W_{ho} h_{t-1} + W_{co} \circ c_t + b_o) \tag{11}$$

$$h_t = o_t \circ \tanh(c_t) \tag{12}$$

where $\circ$ denotes the Hadamard product. i, f, o and c are the input gate's activation vector, forget gate's activation vector, output gate's activation vector and memory cell state vector, respectively. W is a weight matrix to be learned during the training phase.

We employ a combination of both the LSTM and a CNN, known as a ConvLSTM. This allows for the processing of sequential visual data, which in the task at hand corresponds to the weighted glimpses [17]. This can be formulated as, where $*$ denotes the convolution operator:

$$i_t = \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \tag{13}$$

$$f_t = \sigma\big(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f\big) \tag{14}$$

$$c_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \tag{15}$$

$$o_t = \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \tag{16}$$

$$\mathcal{H}_t = o_t \circ \tanh(\mathcal{C}_t) \tag{17}$$

The output of the ConvLSTM is connected to a fully connected layer with a sigmoid activation function. Therefore, the output of the model is a probability value for each of the classes, namely goodware or malware. The loss during the training process is calculated using the binary cross-entropy of the true class and model output, formulated as:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i \times \log(p_i) + (1 - y_i) \times \log(1 - p_i)) \tag{18}$$

3. DATASET

In this paper, we utilize a 32-bit ARM-based IoT application dataset, with 512 samples of malware and goodware presented in opcode format. The dataset was generated by HaddadPajouh *et al.* using the Virus Total Threat Intelligence platform [18]. The class distribution is 52.3% and 47.7% goodware and malware, respectively.

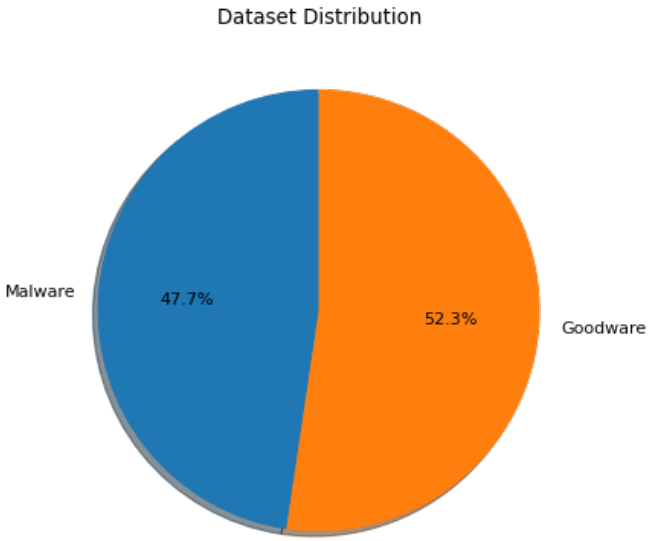

Figure 5; Class Distribution in Dataset

The dataset has a total of 356 unique opcodes which occur at varying frequencies. The frequency count for the ten most frequent opcodes in the entire dataset is presented in Figure 6. Figures 7 and 8 show the ten most frequent opcodes in the respective goodware and malware data batches.

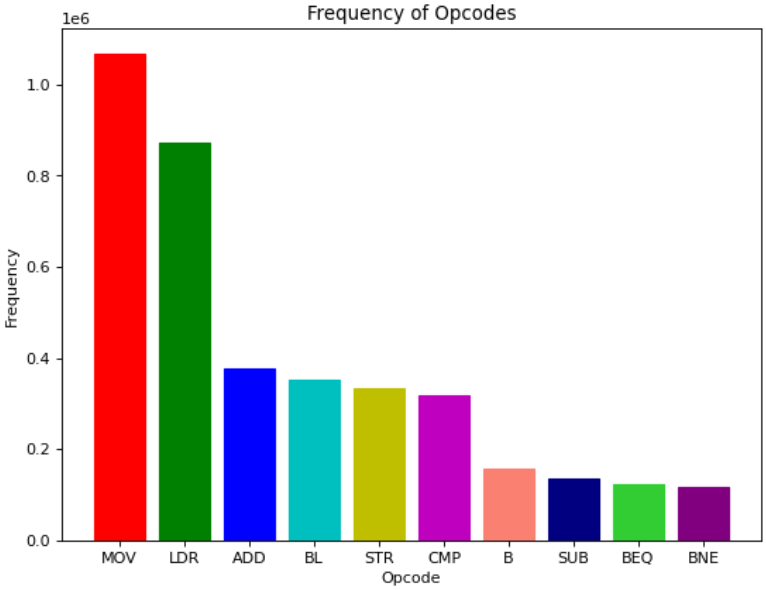

Figure 6: Ten Most Frequent Opcodes in Dataset

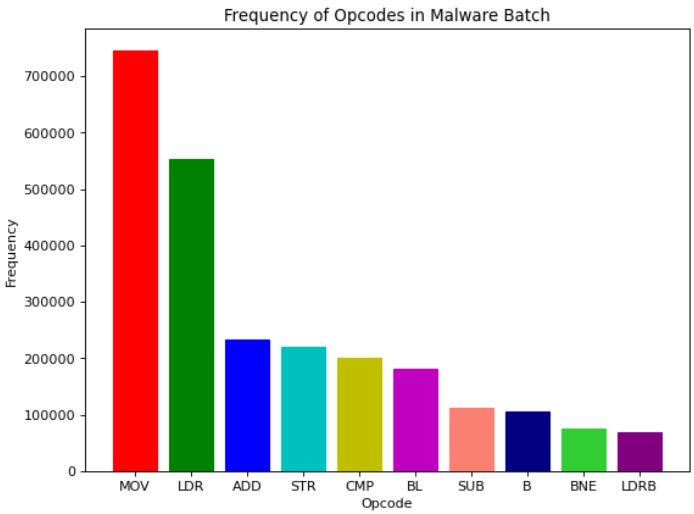

Figure 7: Ten Most Frequent Opcodes in Malware Batch

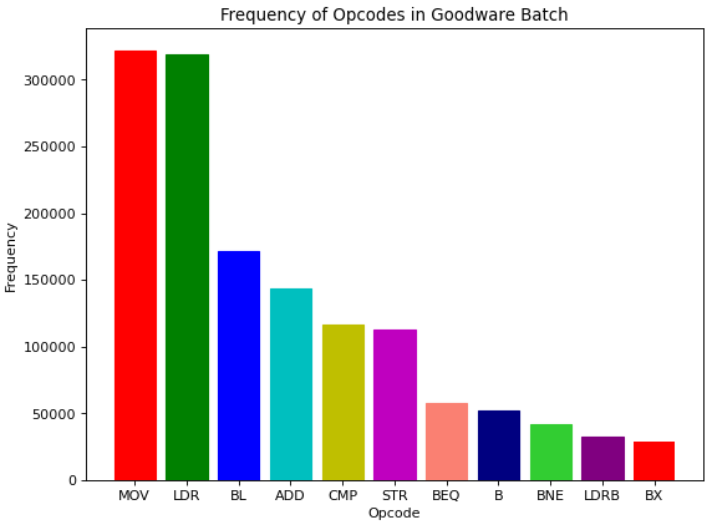

Figure 8: Ten Most Frequent Opcodes in Goodware

4. EXPERIMENT AND DISCUSSION

The proposed method is compared against a simple standalone convolutional neural network which takes the original whole visualized opcode input, rather than glimpses. Both models are trained with a P100 NVIDIA GPU over 100 epochs with Adam gradient optimization using a learning rate of 0.001. An 80-20 train-test split is utilized on the initial 512 data points, with no data augmentation.

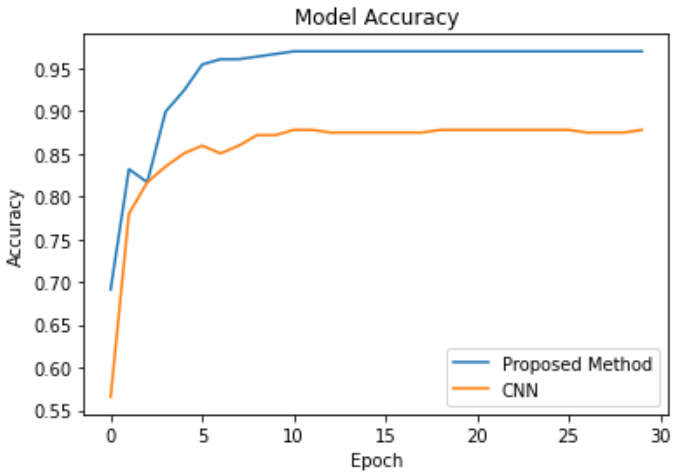

Figure 9: Training Accuracy Over First 30 Epochs

The training results are depicted in Figure 9: Training Accuracy Over First 30 Epochs where the proposed method clearly achieves higher training accuracy than the standalone convolutional neural network. However, it is important to note that the proposed method does take significantly longer to execute than the baseline model. The reported execution time for the proposed method is almost three times longer than the baseline. Figure 9 clearly shows the dramatic increase in execution time for a single classification on an NVIDIA P100 GPU.

Table 1: Training and Testing Results

Approach	Mean Training Accuracy	Mean Testing Accuracy
Proposed Method	**96.665%**	**94.174%**
Convolutional Neural Network	87.33%	86.320%

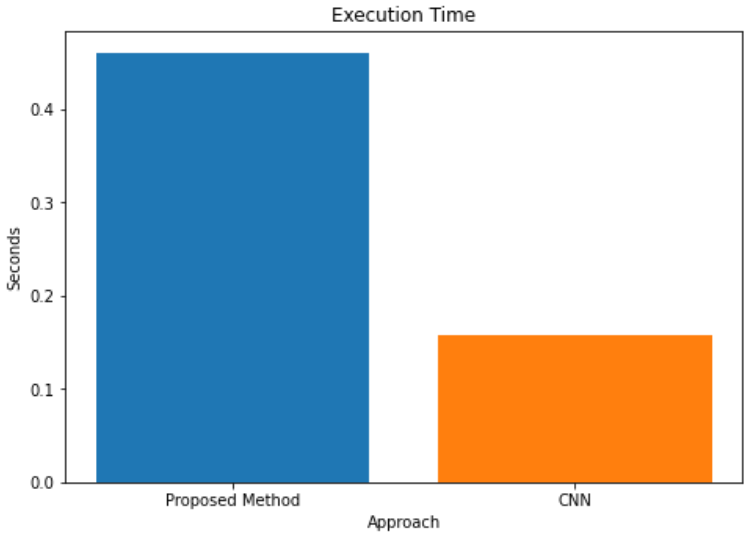

Figure 10: Execution Time

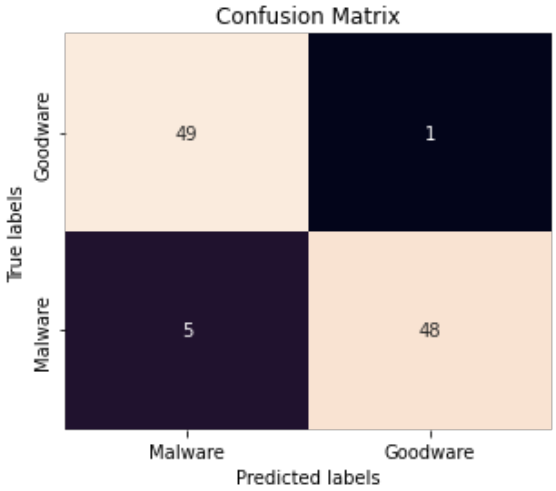

Figure 11: Confusion Matrix

Figure displays a confusion matrix showcasing the results achieved in the testing phase. The results indicate the model was capable of detecting most malware samples with a small number of false positives and negatives. Note that reducing false negatives is especially crucial to malware detection.

Table 2: Performance Metrics

Accuracy	Precision	Recall	F1
94.174%	97.95%	90.56%	94.11%

The results in Table 2 show the aforementioned performance metrics. A high recall is vital to ensuring that malware does not go undetected. The proposed method was able to achieve high precision, however, the recall was lower than all other metrics.

In addition to the comparative analysis conducted with the standalone convolutional neural network, we compare the proposed method against various other machine learning classification algorithms.

Table 3: Comparative Analysis of Variant Classification Techniques

Approach	Accuracy
Proposed Method	**94.174%**
SVM [18]	72.12%
Naïve Bayes [18]	87.51%
MLP [18]	59.07%
KNN [18]	94%
Decision Tree [18]	89.36%

The proposed method outperformed all other algorithms in regard to achieving higher accuracy. However, the increase in accuracy comes with a rise in execution time. While the accuracy certainly justifies the increase in execution time, implementing the proposed approach in practical malware detection applications, which necessitate quick recognition, could have issues. Therefore, more work is needed to maintain model accuracy while reducing computational complexity.

5. CONCLUSION

In this brief paper, a visual attention network is employed for the purpose of visual binary malware detection using program opcode data. The model architecture is composed of three phases: encoding, visualization, and visual attention. The utilization of opcode data allows for a barebones approach for detecting program intention. We utilize visualization to capture fundamental relationships between program opcodes, especially non-adjoining instructions. Furthermore, we employ visual attention to visualized opcode data to identify key regions of focus in aim of enhancing the identification of fundamental program structure and, therein increasing detection of potential polymorphic code. The proposed method is tested against an ARM-based IoT application dataset composed of opcode data. The proposed method is evaluated with various metrics, including accuracy, recall, precision and F-score. In addition, a comparative assessment is conducted on the performance of the proposed model and numerous other models in detecting unseen malware samples. The proposed method outperforms all other compared techniques; however, the execution time increases substantially with implementing the proposed approach. Fundamentally, malicious programs will need to be detected in real-time. Therefore, future work will look at modifications that can be made to the proposed model's structure to maintain model accuracy while reducing execution time.

6. REFERENCES

[1] A. Gupta, P. Kuppili, A. Akella, and P. Barford, "An empirical study of malware evolution," in *2009 First International Communication Systems and Networks and Workshops*, Bangalore, India, Jan. 2009, pp. 1–10. doi: 10.1109/COMSNETS.2009.4808876.

[2] L. S. Tupadha and M. Stamp, "Machine Learning for Malware Evolution Detection," *ArXiv210701627 Cs*, Jul. 2021, Accessed: Mar. 22, 2022. [Online]. Available: http://arxiv.org/abs/2107.01627

[3] A. Sharma and S. K. Sahay, "Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey," *Int. J. Comput. Appl.*, vol. 90, no. 2, pp. 7–11, Mar. 2014, doi: 10.5120/15544-4098.

[4] L. Surya, "An Exploratory Study of Machine Learning and It's Future in the United States," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3782228, Jan. 2016. Accessed: Mar. 22, 2022. [Online]. Available: https://papers.ssrn.com/abstract=3782228

[5] "Machine Learning Market Size, Share and Growth - 2022." https://www.marketsandmarkets.com/Market-Reports/machine-learning-market-263397704.html (accessed Mar. 22, 2022).

[6] M. Kruczkowski and E. Niewiadomska-Szynkiewicz, "Comparative Study of Supervised Learning Methods for Malware Analysis," *J. Telecommun. Inf. Technol.*, vol. nr 4, 2014, Accessed: Mar. 22, 2022. [Online]. Available: http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-13105d82-68da-4a6e-bc1d-3024abc6ffdc

[7] A. Kumar and T. J. Lim, "EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Apr. 2019, pp. 289–294. doi: 10.1109/WF-IoT.2019.8767194.

[8] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for android IoT devices using deep learning," *Ad Hoc Netw.*, vol. 101, p. 102098, Apr. 2020, doi: 10.1016/j.adhoc.2020.102098.

[9] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear SVM-Based Android Malware Detection for Reliable IoT Services," *J. Appl. Math.*, vol. 2014, p. e594501, Sep. 2014, doi: 10.1155/2014/594501.

[10] C.-W. Tien, S.-W. Chen, T. Ban, and S.-Y. Kuo, "Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features," *Digit. Threats Res. Pract.*, vol. 1, no. 1, p. 5:1-5:19, Mar. 2020, doi: 10.1145/3378448.

[11] N. Milosevic, A. Dehghantanha, and K.-K. R. Choo, "Machine learning aided Android malware classification," *Comput. Electr. Eng.*, vol. 61, pp. 266–274, Jul. 2017, doi: 10.1016/j.compeleceng.2017.02.013.

[12] N. Alsadi, H. Karimipour, A. Dehghantanha, and G. Srivastava, "A Recurrent Attention Model for Cyber Attack Classification," in *AI-Enabled Threat Detection and Security Analysis for Industrial IoT*, H. Karimipour and F. Derakhshan, Eds. Cham: Springer International Publishing, 2021, pp. 237–250. doi: 10.1007/978-3-030-76613-9_13.

[13] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 2657–2661. doi: 10.1109/ICASSP.2016.7472159.

[14] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks," *ArXiv170901507 Cs*, May 2019, Accessed: Mar. 22, 2022. [Online]. Available: http://arxiv.org/abs/1709.01507

[15] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[16] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[17] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *ArXiv150604214 Cs*, Sep. 2015, Accessed: Mar. 22, 2022. [Online]. Available: http://arxiv.org/abs/1506.04214

[18] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018, doi: 10.1016/j.future.2018.03.007.