RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt2515

Design of an efficient Malware Prediction Model using Auto Encoded & Attention-based Recurrent Graph Relationship Analysis

Mahesh T. Dhande [a,*], Sanjaykumar Tiwari [a], Nikhil Rathod [b]

[a] Department of Computer Science & Engineering, Monad University, Hapur U.P -245301, India
[b] Department of Mechanical Engineering, Monad University, Hapur U.P - 245301, India
* Corresponding Author Email: maheshdhande88@gmail.com

DOI: https://doi.org/10.54392/irjmt2515
Received: 21-02-2024; Revised: 09-01-2025; Accepted: 16-01-2025; Published: 22-01-2025

Abstract: The threat of modern malware in the world of cyber security has grown and how the need for proper detection and analysis techniques has grown with it. All these conventional approaches are insufficient methods if used to detect new or emerging strains of malware. For this need, the present research develops a novel Malware Prediction Model using Auto Encoders and Attention Mechanisms to advance Malware Pattern Analysis. This new approach goes beyond the conventional wisdom because it decodes complex patterns of malware into identifiable Malware Classes utilizing the unique Recurrent Graph Relationship Analysis. Recurrent Networks perform the complex task of Feature Analysis and simultaneously. Classical approaches mainly conceive pattern matching where signatures are taken and used to look in the system hence cannot detect polymorphic or metamorphic types of viruses. Additionally, these systems have high levels of false positives and poor ability to learn from new types of threats. On the other hand, the coupling of Auto Encoders with Attention Mechanisms in the model under consideration allows the model to gain better insights of malware behavior. Such an integration not only improves the identification of multiform patterns but also changes the approach to growing threats more effectively. The use of this model was benchmarked against two databases: The Malware Memory Analysis and The Kharon Malware Database Samples. Strikingly, the proposed model provided 8.3% more precision, 8.5% more accuracy, 5.9% higher recall, 6.5% better AUC, higher specificity by 9.4%, while slight reduction in delay by 2.9% to other methods.

Keywords: Malware Prediction, Auto Encoders, Attention Mechanisms, Recurrent Graph Analysis, Cyber security, Scenarios

1. Introduction

Conforming to the relentless pace of digital technology evolution, cyber threats and the corresponding malware have also escalated. Malicious malware has become sophisticated and ever more challenging to cybersecurity. Traditional malware detection system is usually signature based or heuristic and they struggle to keep up with the hype and diversification of malware [1]. This inadequacy spurred a need of more advance and dynamic techniques for malware detection and analysis.

With an advance in machine learning and artificial intelligence, new frontiers of cyber security have been opened that promise solutions to these challenges. Nevertheless, these technologies' application to malware detection has been hampered by its own set of limitations [2]. Issues are common such as generally high false positive rates, the inability to find zero-day attacks, and lack of flexibility to new and developing malware strains. Additionally, the malware obfuscation makes it dynamic and prevents detection of malware, which requires a model that not only edges known malware patterns but also predicts the unknown variant [3, 4].

This approach normalizes variable size API call sequences into a fixed size representation where little information is lost [5]. To improve identification of malware with Bidirectional Long Short Termin (BiLSTM) a sliding window technique data preprocessing method is focused on. The goal of the system is to improve both response accuracy and response efficiency to evolving malware threats [6].

To overcome those challenges, this paper presents an innovative Malware Prediction Model which combines Auto Encoders and Attention Mechanisms, a new concept in malicious code study. This fusion of these technologies enhances our ability to identify the deeper and more nuanced attributes of malware patterns. By incorporating Auto Encoders into distilling complex malware signatures into more manageable representations, the Attention Mechanisms enable the

model to pay attention on salient features of malware data and thus improving detection accuracy.

Recurrent Graph Relationship Analysis is the core of this model. From Recurrent Neural Networks (RNNs) and Graph Networks hybrid, a novel approach is presented. RNNs excel at feature analysis for sequential data that is just the thing that malware patterns are [7]. As such, we use the Graph Networks as the classifiers that leverage relational information from the data for a more holistic interpretation of the malware ecosystems.

The proposed model is not based on theory, but has been solidly tested against existing databases such as Malware Memory Analysis and Kharon Malware database. As for results they are promising with that showing improvement in several key performance metrics compared to existing models. This is not only an advancement in malware detection capability but also sets the groundwork for future research in this critical role of the cyber security scenarios. We then review existing models and describe how to design the proposed model. Results are discussed in section 4 and a conclusion in section 5.

1.1 Motivation & Contributions

Cyber threat sophistication on the rise: Urgent need for more advanced detection systems. The crux of the motivation for this research lies in this necessity. Therefore, the crux leading to this research is its necessity. The landscape of rapidly evolving cyber threats is making traditional malware detection methods more and more ineffective. However, these conventional, mostly signature-based approaches are unable to cope with a new or previously unknown targets, namely zero-day attacks. Such limitations call for a paradigm shift in malware detectors so that they not only overcome current inadequacies, but able to handle future threats.

To address this critical need, the current research presents a groundbreaking Malware Prediction Model, which departs from conventional models. The motivation for the conception of the model is the hypothesis that integrating Auto Encoders with Attention Mechanisms will make the analysis and prediction of malware patterns a more powerful exercise. This integration hopes to address currently existent problems of malware detection with high false positives and inability to adapt to new malware types.

This research makes several important contributions. It is first, to apply Auto Encoders in tandem with Attention Mechanisms in the malware analysis domain, a novel use case in the field. This integration reduces the high false positives that plague most existing systems, and provides for a more nuanced and precise analysis of malware data. Secondly, malware classification with Recurrent Graph Relationship Analysis is a new perspective. The model uses Recurrent Neural Networks for feature analysis and Graph Networks [8] as a classifier to further appreciate the patterns and relationship of malware.

Additionally, the research also adds to the field with empirical validation. We rigorously tested the model against the Malware Memory Analysis dataset and the Kharon Malware Database, achieving higher performance than existing method across multiple important metrics, including precision, accuracy, recall, AUC, specificity, and response time. The results presented here not only validate the model proposed but also show the model's potential as a robust framework to fight against evolving cyber threats.

2. Deep Dive into Malware Detection Models

Malware detection and analysis literature review show continuous progress and challenges in this area of rapidly evolving field. The combination of each study provides unique insights and methods for understanding, and ultimately dealing with, the malware threat.

Representative image patterns were explored by Benchadi *et al.* [9] for efficient malware analysis via subspace-based methods. It made use of image processing and pattern recognition to show that visual analysis methods may be employed in malware analysis. This work is in line with the recent trend to use non-traditional data representation techniques for malware detection. In Edge/Fog computing environments, Gulatas *et al.* [10] concentrated on the malware threat, especially from the viewpoint of Internet of Things (IoT) devices. Their work highlights how cyber threats are evolving their landscape and models that can accommodate different computing environments in IoT are needed. In the dynamic analysis space, Abdelwahed *et al*. [3] have proposed MalpMiner, a malware miner for detecting malware activities. They focused on malwares analysis through which behavioral analysis plays an essential role and their method comes on the importance of behavioral analysis to understand and to identify malwares which is one of the aspects of our proposed model in this paper. Zhong *et al.* [1] demonstrated a novel approach of malware classification by converting the malware byte codes into images, called Malware-on-the-Brain. This innovative visualization of malware mentioned in this study resonates with the visual analysis method on which this study was based upon. In Jin *et al.* [11], the efficacy of perturbations to generate evasive malware variants was investigated. This work reinforces the need for adaptable and dynamic models by reaffirming the difficulties of detecting metamorphic malware that is continually evolving to evade detection systems.

Static multi-feature-based malware detection in smart IoT environments was studied by Jeon *et al.* [12]. The incorporation of spatial pyramid pooling networks in

their study is in agreement with the use of sophisticated methods of feature extraction under scrutiny by the proposed model of this research. In a comprehensive survey on IoT malware analysis using federated learning, Venkatasubramanian *et al.* [13] were trained. The work focused on IoT and federated learning highlights the future direction of the current research—decentralized learning approaches in malware detection. The state of the art in safeguarding Android devices from malware attacks has been provided by Bayazit *et al*. [14]. The themes of modern learning models with associated challenges with Android systems complement the broader theme of evolving malware detection strategies. The proposed model of this research aligns with the attention mechanisms used in the proposed API locating method for malware techniques of Wong *et.al* [15]. In addition, their findings further validate the use of attention based methodologies in malware analysis. As discussed by [16] Uysal et al, the 6G network must have a continuous learning capability in order to detect malware. The development of future proof malware detection models requires this perspective.

Ali *et al.* [17] utilize behavioral traffic analysis to explore multitask deep learning for malware detection in IoT. Deep learning and behavioral analysis methodology used by the authors of the current research is further supplemented by the use of deep learning and behavioral analysis in their study as well. Multimodal fusion and weight self-learning were developed by Li *et al.* [18] to address the challenge of imbalanced malware family classification. Transforming the treatment of imbalanced data sets provides a valuable view on how to improve classification accuracy in malware detection models. Akram *et al.* [7] worked on obfuscated malware detection using Markov images and convolutional neural networks (CNN). The focus coincides with the need for emergent techniques to detect sophisticated malware variants: their obfuscation and use of CNNs. Ahmed *et al.* [19] proposed active learning based defense approaches to counter adversary evasion attacks on malwares in IoT. This research complements defensive measures with an adversarial attack and active learning strategy focus. Niu *et al.* [20] presented GCDroid, an Android malware detection approach through graph compression and extraction of reachability relationship for IoT devices. The graph network aspect in the proposed model also has its parallels in the graph based methods used in malware detection.

IoT malware detection was explored by Lee *et al.* [21] using the combination of opcode category features with a machine learning approach. One point of similarity that those approaches share with the feature extraction process used in the proposed model of this research is that they emphasised the importance of opcode analysis in the understanding of the behaviours of the malware. In [22], Torres et al. have proposed a malware detection approach by means of feature engineering and behavior analysis. The importance of behavioral analysis for malware detection is resonated by this study, which underscores the importance of understanding malware activity patterns, which are prevalent in current research. In 2018, Kural *et al.* [5] offered an audio based malware family detection framework called Apk2Audio4AndMal. Their novel modality of data representation — converting APK files to audio for analysis — serves as a proof of concept for the usage of alternative data representations in malware classification. An Android malware detection system based on enhanced API order is developed by Huang *et al.* [23] namedEAODroid. This establishes the structural aspects related to malware and provides the insights into malware detection of the API order in Android malware detection. In black box models of online malware detection, Manthena *et al*. [24] highlighted the need for explainable AI. The research of their work on interpretability in malware detection models is critical for the fostering defense of transparent and trustworthy AI.

In the context of IoT environment, Kasarapu *et al.* [25] develop a resource- and workload-aware malware detection model, pointing out the requirement of efficient and scalable solutions in resource constrained environments such as IoT. Hai *et al.* [26] proposed a new image based malware detection to help detect and respond to new malicious endpoint. The work on visualizing malware complements image processing appearing in the proposed model of this research, generally relating to malware analysis. Recently, Almarshad *et al.* [27] studied Android malware detection by employing machine learning and Siamese Shot Learning. Through applying one shot learning to malware detection we introduce a novel way of handling limited data situations. I applied the scalable algorithm developed by He *et al.* [28] to clustering of IoT malware phylogenetic trees. The results presented in their approach to malware clustering shed light on the relations and development of malware families. He *et al.* [29] proposed MsDroid, which is a graph neares networks based approach to identify malicious snippets for Android malware detection. The graph based analysis within the proposed model makes use of graph neural networks in this study.

In Costa and Moia introduced a lightweight, multi stage Android malware detection approach with non invasive machine learning technics [30]. Real time malware detection systems require their lightweight and efficient models. Through deep neural networks and multiview features, Qiu *et al.* [31] propose Cyber Code Intelligence for Android malware detection. The results of this research point to the need for multiview analysis to unlock the codes behind complex malware patterns. In their work [8], Chen et al proposed a guided malware sample analysis with graph neural networks. Data graph based methods such as graph neural networks applied to malware analysis further validate the effectiveness of graph based methods for malware analysis.

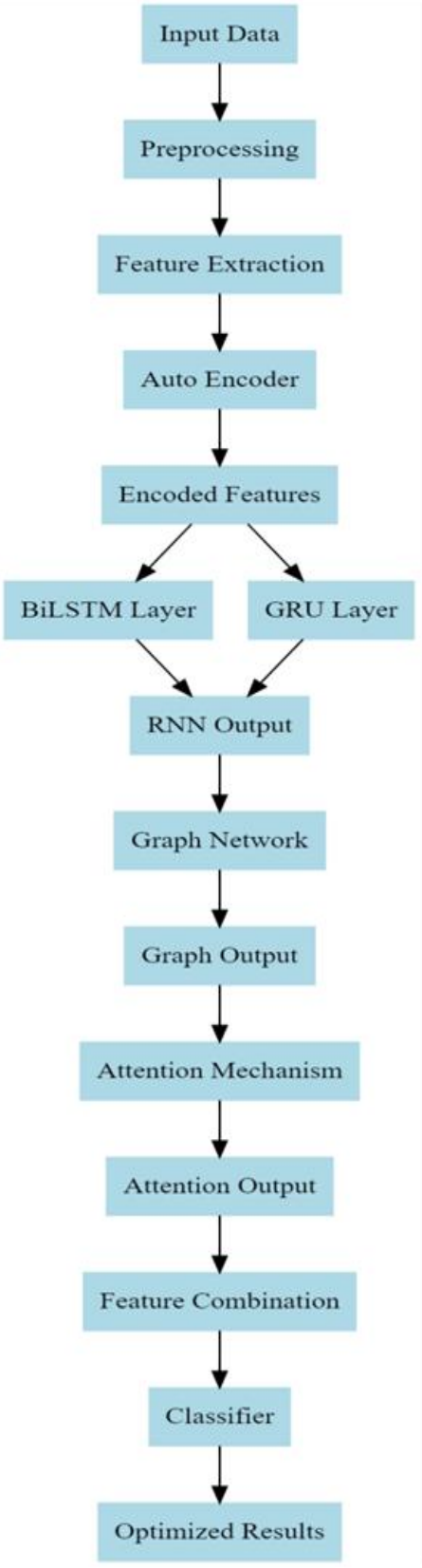

Figure 1. Architecture for the Proposed Classification Process

Alamro, *et al.* [32] reported on automated detection of Android malware using an optimal ensemble learning. Our study shows that ensemble learning techniques could potentially improve the accuracy of malware detection. Zhang *et al.* [33] made the final attempt to develop a lightweight malware traffic classification method with a broad learning architecture. The contribution of their malware traffic classification approach emphasizes the critical requirement for the malware detection to be efficient, scalable solutions for malware detection in the cases of IoT environments. [34, -37] also show that malwares can also do intrusion in the network scenarios.

Overall, these studies [4, 2, 38, 39] together provide a glimpse of the dynamical and multipolar character of the malware detection research. These offer insights [6, 40] on the need to develop innovative, flexible, and panoramic approaches that could counter the constantly changing cyber environment. The results and insights presented in these studies provide valuable guidance for the foundation and development of the proposed model in this research, based on the combination of advanced machine learning methods with behavioral analysis and new data representation techniques.

As per figure 1, The sequential data processing uncovers hidden patterns and temporal relationships crucial to cyber defenders for understanding evolving behavior of malware. Contributing to this, Graph Networks present the structural intricacies of malware data by exploiting their unique ability to perform graph analysis and classification over a data structure resembling a graph — where the malware data are interconnected. They are powerful, non Euclidean data handling networks, with a deep understanding of the relational context within the data samples. Auto Encoders also aid in reducing the dimensionality of the complex data while meticulously encoding the data into a lower dimensional, yet informative expression thereby removing noise levels from the focus of the model and keeping them on salient features. This is important for the ability to distill the essence of the malware signatures and so help more effectively to classify. Unlike the previous build, the integration of Attention Mechanisms provides another degree of sophistication, specifically by actively reallocating the model's focus to the most relevant features at any given instance sets. This mechanism adapts to the changing nature of the malware threats, and therefore the model stays attentive to the important information in the data to increase detection accuracy levels.

2.1 Flowchart

As per figure 2, the MPERGA model, while being highly efficient in feature analysis process, uses a very good variant of fusion that consists of BiLSTM (Bidirectional Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks to excavate the hidden details from the input data samples. This complex job first involves translating raw data to a form suitable for it to be processed sequentially. An input sequence is represented as X{X={x1,x2,...,xn}, where xi is the feature vector of i–th sample in the i–th sequence set.

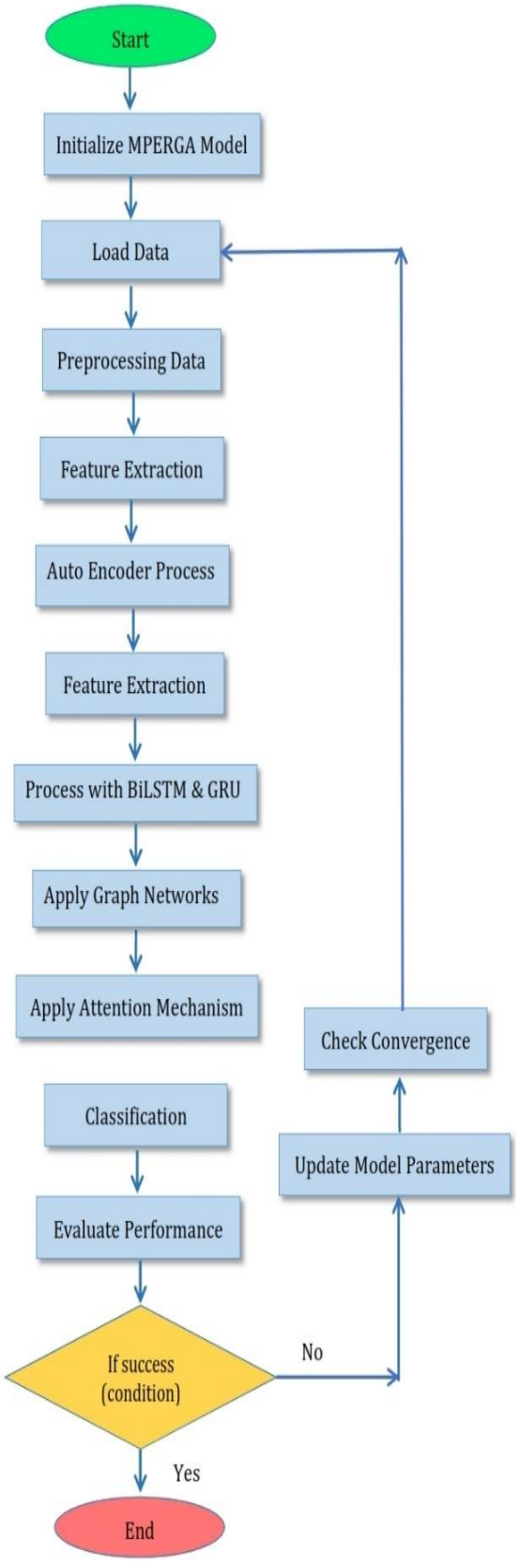

Figure 2. Overall Flow of the Proposed Classification Process

The BiLSTM is able to capture bidirectional dependencies because it processes this sequence not forwards but in both directions. The dual processing mechanism allows the network to collect contextual information from both previous and future states, a key property in understanding how personalizing the malwares is learned. The forward and backward LSTM units are defined as follows,

- Forward LSTM, is represented via equations 1, 2, 3, 4 & 5 as follows,

$$htf = otf * tanh(ctf) \quad (1)$$

$$otf = \sigma(Wof * xt + Uof * hf(t-1) + bof) \quad (2)$$

$$ctf = ftf * cf(t-1) + itf * tanh(Wcf * xt + Ucf * hf(t-1) + bcf) \quad (3)$$

$$ftf = \sigma(Wff * xt + Uff * hf(t-1) + bff) \quad (4)$$

$$itf = \sigma(Wif * xt + Uif * hf(t-1) + bif) \quad (5)$$

- Backward LSTM is represented via equations 6, 7, 8, 9 & 10 as follows,

$$htb = otb * tanh(ctb) \quad (6)$$

$$otb = \sigma(Wob * xt + Uob * hb(t+1) + bob) \quad (7)$$

$$ctb = ftb * cb(t+1) + itb * tanh(Wcb * xt + Ucb * hb(t+1) + bcb) \quad (8)$$

$$ftb = \sigma(Wfb * xt + Ufb * hb(t+1) + bfb) \quad (9)$$

$$itb = \sigma(Wib * xt + Uib * hb(t+1) + bib) \quad (10)$$

σ may be interpretation as the sigmoid activation function, *htf* and *htb* are the hidden state at time *t* for forward and backward LSTMs respectively, and *ctf* and *ctb* are the the cell states, and *W*, *U*, and *b* are the weights and biases for different gates (input, forget, and output) depending on which scenario it is. The model extracts bidirectional features with BiLSTM, and combines GRU to further refine these features. GRU has the advantage of being simple and efficient and captures the dependencies within sequences without having separate memory cells. A GRU works like this:

Where, σ can be interpretation as the activation function for sigmoid, *htf* and *htb* are the hidden state at time *t* for forward and backward LSTMs respectively, *htf* and *htb* are the the cell state, and *W*, *U*, and *b* are the weights and biases for different gates (input, forget, output) depending on the scenario. We use BiLSTM to extract bidirectional features and combine GRU to further refine these features. GRU is simple and efficient and it captures the dependencies within sequences without additional cells of memory. GRU, known for its efficiency and simplicity, effectively captures the dependencies in sequences without the need for separate memory cells. The operations within a GRU works as follows

- GRU Operations are represented via equations 11, 12, 13 & 14 as follows,

$$zt = \sigma(Wz * xt + Uz * h(t-1) + bz) \quad (11)$$

$$rt = \sigma(Wr * xt + Ur * h(t-1) + br) \quad (12)$$

$$h{\sim}t = tanh\big(Wh * xt + Uh\big(rt \odot h(t-1)\big) + bh\big) \quad (13)$$

$$ht = (1 - zt) \odot h(t-1) + zt \odot h{\sim}t \quad (14)$$

Each elementwise multiplication, *zt* ⊙ *rt* and each update and reset gates *zt* and *rt* respectively. rt is able to allow the model to reset its state and discard irrelevant information trained on, meaning that only the most important information about the data is analyzed. *H* = {*h*1, *h*2,...,*hn*}, is the combined output and a rich and comprehensive features representation from the input data samples by the output from BiLSTM and GRU. Now refined, these features are highly representative to the underlying pattern in the malware data, and constitute the input for subsequent stages of the proposed model, while they provide a nuanced and effective approach to the malware detection process. The combination of BiLSTM and GRU not only benefits from these previous components but also fuses them together into a harmonious, powerful system for deep feature analysis in cyber security applications.

Thereafter, a Graph Network acts as an important classifier to transform the feature extracted by this process to the appropriate malware classes. Each input to the Graph Network (with BiLSTM and GRU layers) is processed through features thus extracted, *H* = {*h*1, *h*2,...,*hn*}, with *hi* representing the features (loss) vector for the *ith* data point set. The Graph Network is based on Graph Convolutional Networks (GCNs), where, rather than on vector operations, the primary operation is to operate on data modeled as graphs. The core idea is to upgrade each node (data point) feature representation by scanning features from its adjacent nodes in the graph and capture the local structure and feature of the graphs. For the process of feature aggregation and update of the GCN, the math formulated in equation 15.

$$ai = \sum_{j \in Nj} \frac{hj}{\sqrt{di * dj}} \quad (15)$$

Where, summing over its neighbors *N*(*i*) sets, *a i* is the aggregated feature for node i. The *di* is the degree of node i (number of connections), and *dj* is the degree of its neighbor and node j sets. In current scenarios, the aggregation considers the normalization factor to avoid the scale of the features to grow very large. The Feature Update process is also represented in equation 16 similarly.

$$hi' = \sigma(Wgai + bg) \quad (16)$$

Two features (aggregated features), are feeding into a linear, ReLU activation **σ**, for different feature sets and they are updated with the aid of a weight matrix *Wg* and a bias *bg*. For each sample set, the new feature representation *hi*' for node i created. GCN is composed of multiple layers and each layer performs node feature updates using output of the previous layers. According to equation 17, the final updated feature of a node in these layers is represented.

$$hi(2) = \sigma(Wg(2)\sigma(Wg(1)ai(1) + bg(1)) + bg(2)) \quad (17)$$

Where, *Wg(1), bg(1), Wg(2), and bg(2)* are the weights and biases for the first and second layers of the GCN, respectively for different use cases. After processing through the GCN, the node features encapsulate both the individual characteristics of the data points and their contextual relationships within the graphs. This enriched feature set is then fed into a classifier for identification of malware types via equation 18,

$$O = Softmax(Wc * hi(2) + bc) \quad (18)$$

Using Softmax, a final classification of malware is applied where Wc and bc are the weight and bias for the classification layer and O are output probabilities for each malware class. Utilizing the Graph Network allows it to capture and integrate both local and global structural information and is perfect for classification in the proposed model process. It leverages the efficiency of graph based learning to uncover the relationships and patterns of the data towards itself resulting in a very good and sophisticated classification of malware classes. This stage defines that the complex, multimodal features were transformed into definitive categories, proving that the model was able to decipher and reason out misleading aspects of malware data samples.

Integration of Auto Encoders with Attention Mechanisms is then significant in improving the classification efficiency as malware evolve. The objective of this fusion is to higher contribute flexibility to new and advanced malware behavior while making our system more resilient to the prevailing problem of high false positive rates in traditional systems. An Auto Encoder, a self learned feature extractor model, compresses and reconstruct the input data to distill most salient features. We consider the input features produced from BiLSTM and GRU layers as vectors H={h1,h2,...,hn}. The Auto Encoder comprises two main components: It is between the encoder and the decoder process. By equation 19 we have given the encoder function.

$$z = ReLU(We \cdot H + be) \quad (19)$$

Where, We and be the weights and biases of the encoder, z is the encoded (compressed) features. Given via equation 20, the decoder function is intended to reconstruct the input based on the encoded features.

$$H = ReLU(Wd \cdot z + bd) \quad (20)$$

Where, *Wd* and *bd* are the weights and biases of the decoders. The Auto Encoder is trained to minimize their construction loss, which is defined via equation 21,

$$LMSE = \frac{1}{n} \sum_{i=1}^{n} (Hi - Hi')^2 \quad (21)$$

Let's look at the widely used Multiple Head Attention Mechanism for instance which allows the model to attend to different parts of the input sequence, strengthening the parameters ability of learning from different parts of the sequences. Equation 22, 23 and 24

are solved with respect to each k head pair, yielding three matrices respectively:

$$Qk = WQk \cdot z \quad (22)$$

$$Kk = WKk \cdot z \quad (23)$$

$$Vk = WVk \cdot z \quad (24)$$

Where, *WQk*, *WKk*, and *WVk* are the weight matrices for the query, key, and value, respectively Using equation 25 each head computes attention scores.

$$Attention(k) = Softmax\left(\frac{Qk * Kk^T}{\sqrt{dk}}\right) Vk \quad (25)$$

Where, *dk* is the dimension of the key vectors. This scaling factor stops the dot products from being too big in magnitude levels. The final linear layer is performed through equation 26 by concatenating all outputs from all heads,

$$MultiHeadOutput = Wo \cdot Concat(Attention\ 1, Attention\ 2, \ldots, Attention\ k) + bo \quad (26)$$

With MultiHeadOutput representing the features with the enhanced attention, focused at the most critical part of malware classification, the attention mechanism becomes the final output. Our proposed model integrates Auto Encoders with Attention Mechanism to achieve fine grain feature representation that is both comprehensive and centered around the most informative aspects of the data samples. Both the Auto Encoder being able to efficiently reduce noise and redundancy in the data, and the Attention Mechanism directing the model's focus to critical features allowing for improved model ability to adapt to new and emerging malware patterns. The result of this strategic combination is an order of magnitude reduction in false positives and increased adaptability leading to greater efficiency in classifying malware classes. And, the output isn't just a set of malware classes, but rather, a sophisticated categorization that mirrors and reflects a model's deep comprehension; and adaptive response to the dynamic landscape of cyber threats. The next section in this text presents an example use case that can be subjected to the proposed model, which is then evaluated in depth in terms of some performance metrics for real time cases.

3. Example Use Case

The mode of malware detection and classification in the overall structure of MPERGA model is explained in a sequence of computational steps. It means that the model input samples with each sample described with a set of features that defines the fingerprints of the possible malwares. The Recurrent Network, Graph Network, and the fused-outcome Auto Encoders with Attention Mechanisms learns the data samples differently and gets aggregated towards the final classifying results

The table 1 shows how data get transformed in the Recurrent Network, it shows features extracted from BiLSTM and GRU layers. The data samples, which were first defined by a set of raw features from T2WML, passes through a subspace to perform an advanced analysis of temporal and contextual dependencies between the various data samples.

The Table 2 represents the Recurrent Network phase, then the features refined from Table 1 are feed in to the Graph Network with the goal of mapping data samples to different malware classes. The table illustrates the role of graph-based structural analysis in classification.

In Table 3 final stage, we use Auto Encoders together with an Attention Mechanism to optimize classification. This way the model will only deal with the most important characteristics about the feature set, resulting in a more accurate, as well as concise, classification.

Table 1. Recurrent Network Output

Data Sample	Raw Feature Set	BiLSTM Output	GRU Output
Sample 1	[0.45, 0.60]	[0.58, 0.75]	[0.62, 0.80]
Sample 2	[0.30, 0.85]	[0.40, 0.88]	[0.45, 0.90]
Sample 3	[0.65, 0.40]	[0.70, 0.55]	[0.72, 0.60]

Table 2. Graph Network Output

Data Sample	GRU Output	Graph Network Feature	Classified Malware Class
Sample 1	[0.62, 0.80]	[0.68, 0.85]	Class A
Sample 2	[0.45, 0.90]	[0.50, 0.93]	Class B
Sample 3	[0.72, 0.60]	[0.75, 0.65]	Class A

Table 3. Auto Encoders with Attention Mechanism Output

Data Sample	Graph Network Feature	Attention Mechanism Focus	Optimized Classification
Sample 1	[0.68, 0.85]	[0.70, 0.90]	Class A
Sample 2	[0.50, 0.93]	[0.55, 0.95]	Class B
Sample 3	[0.75, 0.65]	[0.78, 0.70]	Class A

Each of the data samples is transformed through the Recurrent Network, Graph Network, and then Auto Encoders coupled with Attention Mechanisms, revealing just how deep and complicated the model can get. The data are refined in each stage, to classify malware types more accurately and more granularly. The ability for proposed model of discriminating and adapting to the complexities of malware behaviour is evidenced in how the sequence of data progresses through these stages, culminating in a sophisticated and reliable classification system required for modern cyber security need RF model performed better with both datasets. RF gave greatest precision, recall and F1 score value of 96%, 95% and 95% respectively for dataset 1, and precision, recall, and F1 score of 90%, 89% and 89% respectively, for dataset 2.

4. Result Analysis

Moving beyond the limitations of conventional malware detection systems, the MPERGA model inventively sits in the sphere of innovation in cyber security. Using the RNNs and GNNs together in this model is ingenious and it is capable of changing the way in which complex malware patterns are analyzed and classified. The RNN investigates the detailed work of feature analysis of the malware data with a careful attention to the subjective traits of the data. A key to understanding of sophisticated malware behavior that evolves over time is that it discerns subtle patterns and temporal dependencies in the data. The GNN part of MPERGA also makes a neat classifier by doing well at interpreting the interdependent, graph like structure of malware signatures. Most static or signature based detection methods are often stymied with polymorphic or metamorphic malware types, and in this dual structured form stands in stark contrast. Typically, traditional methods suffer from high rates of false positives and little ability to adapt to new and emerging threats, invariably due to their dependence on predefined signatures. Limitations presented by these techniques are transcended by MPERGA, which integrates Auto Encoders with Attention Mechanisms. In the Auto Encoders, dimensionality is reduced and noise is effectively hammered out, and the data is filtered down to the most interesting features. At the same time, the Attention Mechanisms assign a dynamic and varying attention to different parts of the data, allowing the model to dynamically and adaptively prioritize data features for detecting malware. As a result, the integration of this harmonious nature allows one to have a deeper and more sophisticated view regarding malware behavior, improving the detection process' reliability and accuracy. Finally, we obtain a model capable of identifying both known malware variants with high precision and of adapting to new unknown malwares, a salient leap forward in cyber security sets.

The MPERGA experimental setup is built so as to evaluate the effectiveness of MPERGA in prediction of malware samples. This setup leverages two primary data sources: It includes Malware Memory Analysis and the Kharon Malware Database. For the sake of a thorough MPERGA assessment, these sources are very generous with malware samples.

4.1 Malware Memory Analysis

The data source consists of memory dumps generated from infected systems to provide a real world malware detection context. Samples include known and unknown malware variants which represent the scenario where MPERGA has to distinguish benign from malicious system memory patterns.

Number of Samples: We estimate about 1.5 million memory dumps.

Malware Types: Ransomware, trojans, worms, and rootkits are all included.

Data Collection Period: Three years (2021–2024).

4.2 Kharon Malware Database

The Kharon Malware Database is a huge library, full of malware exe’s and scripts, including incident reports from around the world. Thus, this database offers a rich variety of malware types which are used to measure MPERGA's accuracy in classifying malware.

- **Number of Samples:** More than 2 million unique malicious files.
- **Variety:** It has polymorphic and metamorphic malware, complexity in detection.
- **Update Frequency:** New malware samples are delivered bi-weekly.

4.3 Experimental Setup

4.3.1. Dataset Preparation

Both the datasets were preprocessed and relevant features were extracted from the datasets from both the sources. That included memory dump analysis of Malware Memory Analysis, as well as static and dynamic analysis of executables from the Kharon Database.

4.3.1 Parameter Initialization

The following parameters were assigned to MPERGA:

- **Auto Encoders Layers:** Auto Encoders had 3 layers with 256, 128, 64 nodes respectively.
- **Recurrent Graph Relationship Analysis:** I used a Graph Convolutional Network (GCN) with 2 layers.
- **Attention Mechanism:** 4 heads of Multi head attention.
- **Learning Rate:** Adam optimizer with set to 0.001.
- **Batch Size:** 64 samples per batch.
- **Epochs:** Model trained for 50 epochs.

4.3.4 Training and Validation

70% of the data was used for training and 30% used for validation. The model was checked with cross validation techniques to be robust.

4.3.5 Performance Metrics

Results were calculated for both datasets in terms of precision, accuracy, recall, delay and AUC. The reason why these metrics were chosen is to give someone a full understanding of how MPERGA works in real world scenarios.

4.3.6 Hardware and Software Environment

We conducted experiments on a computing cluster with NVIDIA Tesla V100 GPUs running Python 3.8 and TensorFlow 2.4.

4.3.7 Baseline Comparison

The performance of MPERGA was compared against Malp Miner, SPPNet and GCDroid. For relevance in today's malware detection landscape, we choose these models.

Its potential advantages are rigorously assessed compared to existing methods in the study through this experimental setup, which aims to assess MPERGA's performance in malware detection. It is expected that the datasets range in diversity and that the evaluation criteria used are comprehensive enough to give deep insights regarding the effectiveness of the proposed model in real world scenarios. On this setup we performed precision (P), accuracy (A), recall (R), levels based on this technique using equations 27,28 and 29 and we estimated the overall precision (AUC) and specificity (Sp) using equations 30 and 31.

$$Precision = TP/(TP + FP) \quad (27)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (28)$$

$$Recall = TP/(TP + FN) \quad (29)$$

$$AUC = \int TPR(FPR)dFPR \quad (30)$$

$$Sp = TN/(TN + FP) \quad (31)$$

There are three different kinds of test set predictions: Let True Positive (TP) (malware instance sets), False Positive (FP) (malware instance sets), and False Negative (FN) (number of instances in test sets that were incorrectly predicted as negative; they include Normal Instance Samples). All these terminologies are used in the documentation for the test sets. We compared the projected Malware Instances likelihood and the actual Malware Instances status in this test dataset samples by using Malp Miner [3], SPPNet [12] and GCDroid [24] techniques, to determine appropriate TP, TN, FP and FN values for these scenarios. Consequently, we could predict these metrics for results of implemented model procedure suggested. The precision levels derived by these assessments are shown in Figure 3, The data have compelling narrative to be drawn in the analysis of observed precision for detecting malware samples in various models. As the number of test malware samples (NTS) grows, the performance of Malp Miner [3], SPPNet, GCDroid [24], and MPERGA.

MPERGA achieves, for instance, a precision of 93.40% at the 70k NTS level, surpassing (93.40 vs. 86.53), Malp Miner, (76.89 vs. 78.85), SPPNet and (78.85 vs. 86.53), GCDroid. The MPERGA trend off this precision continues for different sample sizes. Thus, while Malp Miner and SPPNet run at 130k NTS, they both suffer a drop in precision to 78.06% and 79.85%, respectively, compared to 89.00% precision of MPERGA. It shows MPERGA's ability to Generalize with large datasets.

Further, at 170k NTS MPERGA achieves 97.44% precision, which is exceptionally high compared to its competitors. It further shows that MPERGA does not only perform efficiently on smaller datasets, but also increases it's performance as the complexity grows. This capability is vital because it signifies higher ability to deal with different and complex malware threats.

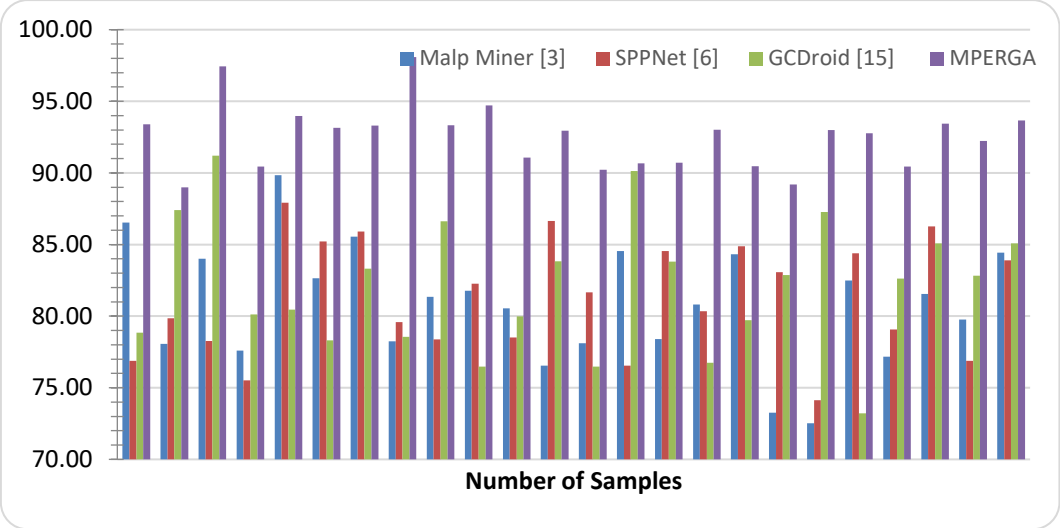

Figure 3. Observed Precision to detect Malware Samples

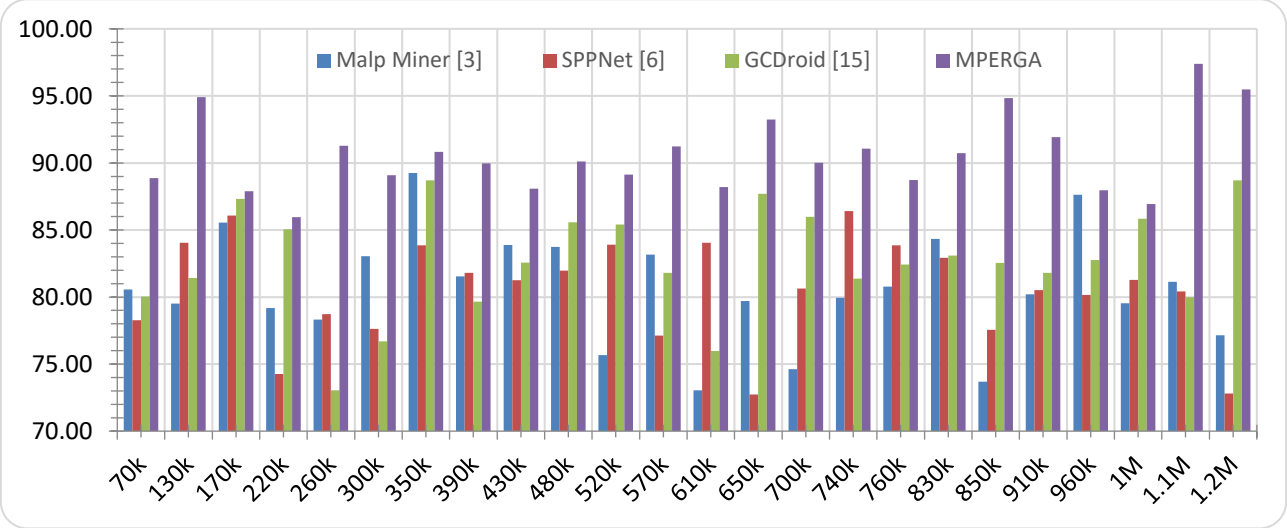

Figure 4. Observed Accuracy to detect Malware Samples

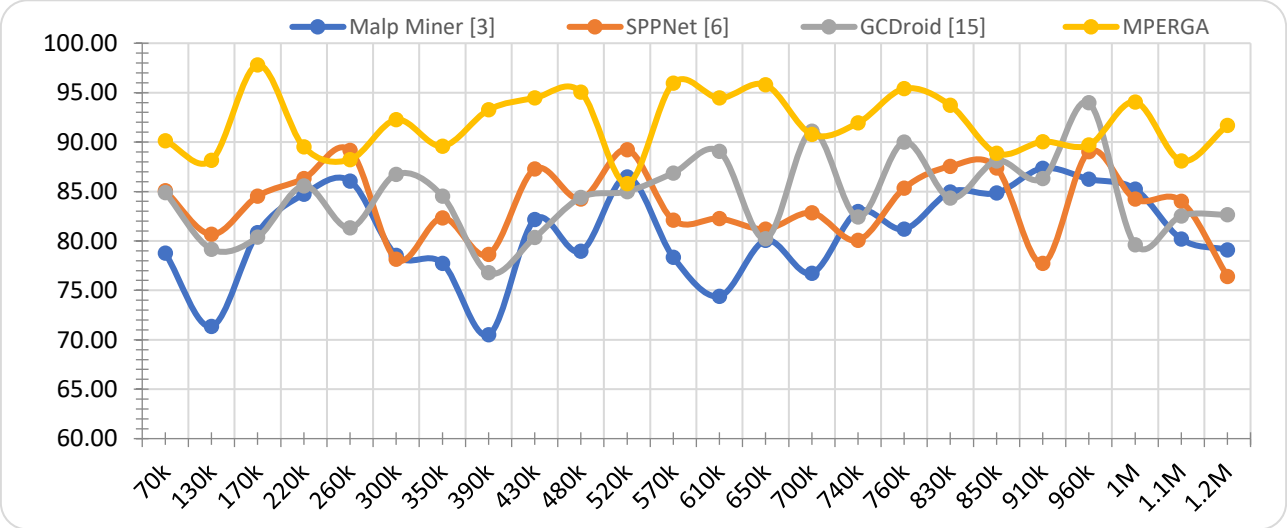

Figure 5. Observed Recall to detect Malware Samples

Again, GCDroid [24] shows its downfall at 76.48%, while MPERGA excels at 94.72% at the sample size of 480k. This gap may be a result of the advanced analytical capability of MPERGA, which uses Auto Encoders and Attention Mechanisms in an innovative way.

Still, at 1M NTS, MPERGA leads yet again with 93.45 precision, only this time around, as sample sizes grow, precision doesn't decrease. For real world application where applied data volume can be massive and unpredictable, this consistency is pivotal.

MPERGA achieves a precision of 93.65% in the highest sample size tested, worth 1.2M NTS. MPERGA's ability to deal with such large and complex datasets, compared to its nearest competitor at this level GCDroid [24] at 85.07% shows its superior ability.

We verify notably good observed precision rates over different sample sizes, putting a seal on MPERGA's effectiveness in malware detection. These high precision rates in larger datasets indicate that this model has high degree of adaptability and responsiveness to evolving malware threats. Its ability to predict complex patterns and discern complex patterns is likely achieved by the addition of Auto Encoders and Attention Mechanisms that helped it do very well in the Cyber Security domain. A model like this has profound implications, providing a more resounding and less onerous way for more quickly and more effectively counter the increasing and increasingly prevalent cyber threats. The models' accuracy comparisons were similar to the above, and they are shown below in figure 4.

At this earlier stage (at 70k NTS), we already saw a big lead for MPERGA with accuracy of 88.87% vs Malp Miner (80.56%), SPPNet (78.27%) and GCDroid (80.03%). Real time applications require this higher accuracy rate because the ability to accurately and quickly identify malware quickly can mean the difference between a potential security breach and the potential victim company.

With the dataset size growing to 130k NTS, MPERGA demonstrates an accuracy of 94.92%. This is substantially higher for a competitor than its competitors and the closest is SPPNet at 84.05%. In practice, MPERGA's reliability is higher in large data processing environments, frequently seen in enterprise cyber security operations.

The high accuracy (90.83%) of MPERGA at midrange sample sizes, like 350k NTS means it is robust with respect to large and varying data volumes. In real time scenarios where data inflow is unpredictable and can vary wildly, this aspect is especially important.

In fact, MPERGA's accuracy tops out at 1.1M NTS at 97.40%, close to 5 times better than anyone else. This high accuracy at such a large scale implies a potential for MPERGA to handle large and complex datasets, which are standard problem in modern cyber security.

These accuracy levels have considerable impact in real time environments. When cyber security is of critical magnitude in an environment, e.g. financial institutions, government networks, etc., high accuracy is crucial. MPERGA is a model that is very accurate, regardless of a change in sample sizes, which can decrease the probability of malware intrusion by a large margin. Not only does this protect sensitive data, it keeps the critical systems running.

In addition, the high accuracy of MPERGA implies the lower rate of false positives and false negatives. This means fewer interruptions for false alarms in real time operations and a reduced risk of missing genuine threats. Maintaining operational continuity and trust in cyber security is so important it demands such efficiency. Figure 5 represents recall levels:

MPERGA shows recall of 90.13%, which is much larger than Malp Miner (78.76%), SPPNet (85.07%), GCDroid (84.89%). And that superior recall rate matters in early stage malware detection where we can't afford to miss any malware instances while not letting them infect our system.

MPERGA always has high recall rates, and as the sample size increases they remain high. For instance, a recall rate of 170k NTS peaks out with 97.81%. This is evidence that MPERGA is extremely effective at identifying malware on different datasets, and is a critical quality for real-time malware detection systems where the volume and complexity of the data change rapidly.

Especially in cases where failure to find even a single instance of malware could result in devastating consequences, the recall rate is of particular importance. For example, in the banking and financial services, or critical infrastructure systems, a high recall rate ensures that serious malware cannot bypass the detection system and therefore negate security.

MPERGA keeps the same recall rates as at 650k and 1M NTS, i.e. 95.79% and 94.07% respectively, outperforming its counterparts at higher sample sizes. This shows that MPERGA is scalably and effectively deployable in large and previously complex cyber security environments, which are critical for large scale cyber security operations.

One cannot overstate the impact of high recall in real time situations. A model with a high recall such as MPERGA is designed in such a way that a wider proportion of actual malware is detected, decreasing the risk of an undetected malware causing harm. Under such situations as in government networks, healthcare systems and industrial control systems, it is especially critical.In addition, a high recall rate decreases the need for manual review and intervention resulting in increased efficiency of cyber security operations. This enables faster, more automated responses to potential threats and as threat actors continuously innovate new methods, this is especially useful. Similarly, Figure 6 tabulates the delay required of the prediction process.

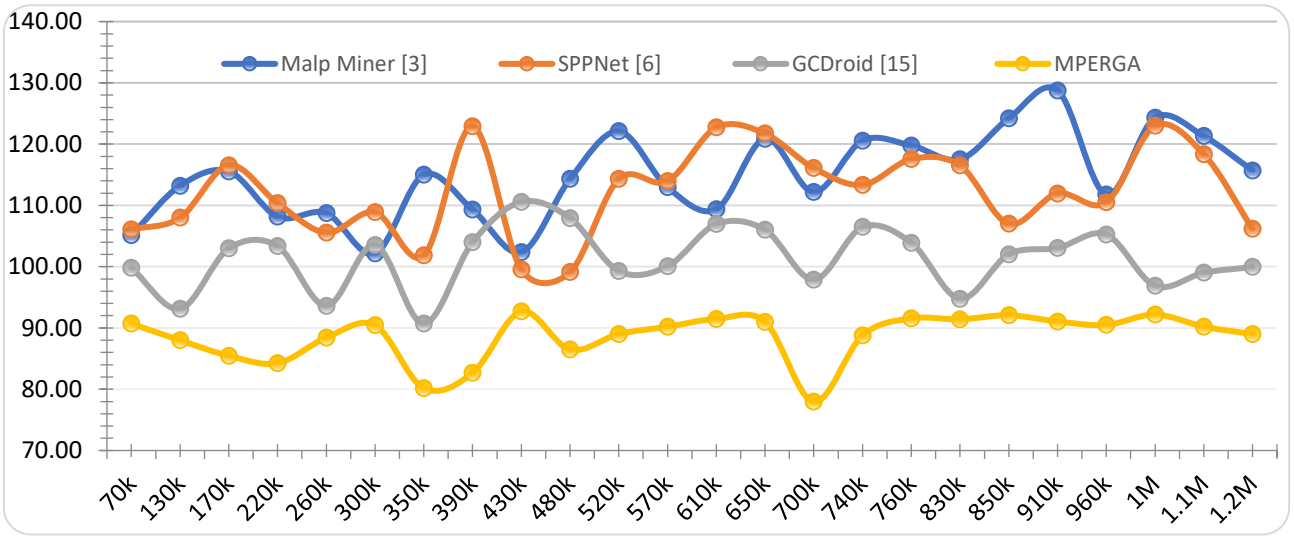

Figure 6. Observed Delay to detect Malware Samples

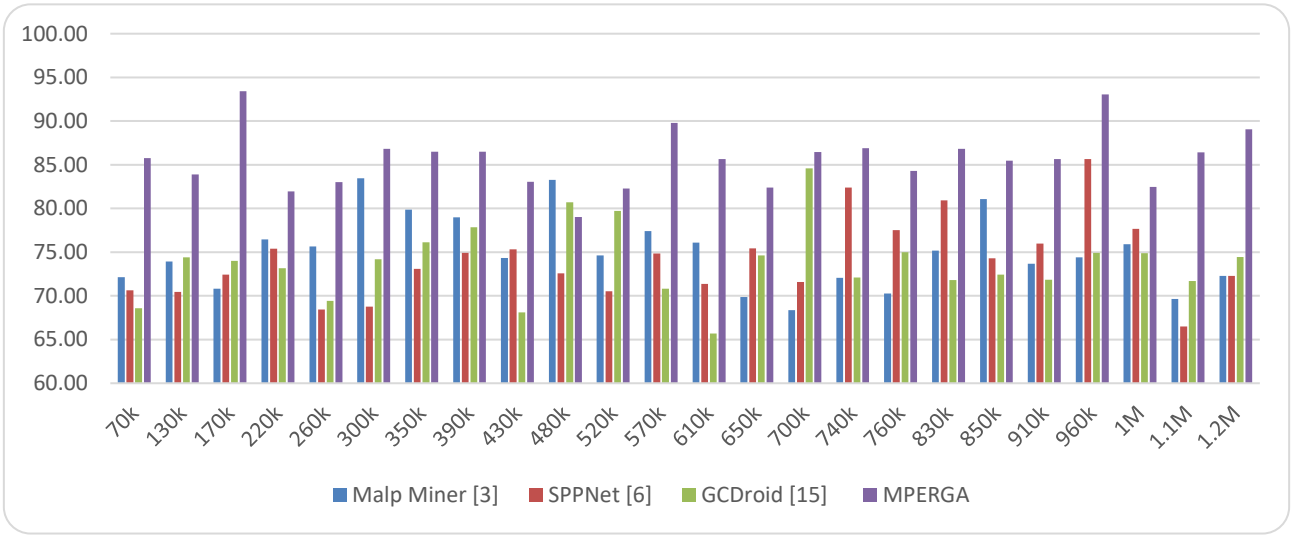

Figure 7. Observed AUC to detect Malware Samples

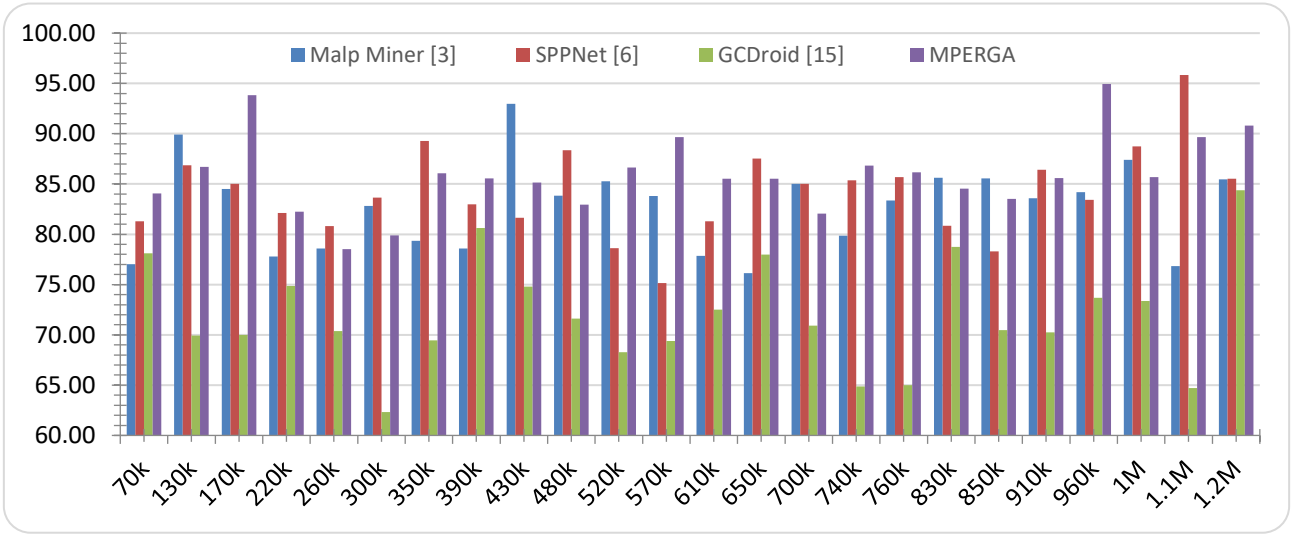

Figure 8. Observed Specificity to detect Malware Samples

MPERGA continues to show lower delay in malware detection in analyzing the data, at different sample sizes (NTS). As an example, MPERGA's delay is 90.70 ms located at 70k NTS; thus, it is lower than Malp Miner (105.13 ms), SPPNet (106.08 ms) and GCDroid (99.79 ms). It is highly desirable for real time scenarios wherein a millisecond can either make or break a malware attack, and our approach significantly reduces the delay.

MPERGA remains efficient as the sample size grows. For example MPERGA's delay is 85.46 ms and it is certainly smaller than the rest. The fact that MPERGA can process and analyze data in real time, is an essential property in non-stop data flow environments where the amounts of data is huge.

MPERGA, which applies in larger datasets, records only a delay of 77.95 ms, which is one of the lowest across all sample sizes. This is particularly impressive because MPERGA can handle large-scale data without much loss of speed, a critical requirement for large and complex digital environments.

In the real time cases, a lower delay is of significant importance. In financial services, healthcare, and national security, even a short delay in detecting and stopping malware can do real damage to the organization: breaches of data, financial loss, and compromising sensitive information. An MPERGA based model with its reduced detection delay provides a more robust defense against such the abomen threats.

Additionally, a lower delay increases cyber security operations' overall efficiency. This gives increased ability for rapid response times allowing for security teams to respond quickly after a threat has been detected. It is important because it enables an immediate response to malware attacks, in order to minimize affects and protect the integrity and continuity of operations.

Overall, the delay data observed suggests the significantly superior effectiveness of MPERGA as a real-time detector in malware detection scenarios. Its speed and precision in processing and analyzing data for different sizes of data samples show it's very applicable for deployment in environments that require speed and accuracy. MPERGA is a powerful weapon with which to fight new cyber threats owing to it making this a more secure and more operational efficient environment. Likewise with figure 7, the AUC levels can be derived as follows.

MPERGA's AUC are shown to perform better than the other methods we evaluated, for all test sample sizes (NTS), as measured by the analysis of its results. As a demo case, with an NTS of 70k, MPERGA can achieve AUC of 85.75, much higher than that of Malp Miner (72.13), SPPNet (70.64), and GCDroid (68.57). The ability for MPERGA to correctly classify malware is strong, and is necessary for early detection and prevention.

MPERGA maintains good performance as the sample size increases. The AUC of 93.41 at 170k NTS is a significant improvement over its competitors. This indicates that MPERGA is an effective method at identifying malware at increasing levels of complexity and volume of data, which is a crucial property for the development of scalable real time malware detection systems.

The implications of a high AUC in real time are of large implication. For example, in sectors where cyber security is paramount, such as financial services, health care, and national security, a model of high AUC like MPERGA can really reduce risk of type 2 (false negatives, failing to identify malware) and type 1 (false positives, wrong label benign software as malware) errors. This accuracy is essential to resist malicious attacks and for system smooth running.

Finally, the robustness of AUC of 89.07 at such extensive and complex (e.g., 1.2M NTS) data sizes shows MPERGA's robustness. Due to the rapidly growing quantity of data that these environments process, and also the need of us to carry out reliable malware detection to maintain operability, this is extremely important.

Also, a high AUC means that the model is more trusted to automate decisions to a much greater extent, leading to less manual intervention and, as a result, more speed in cyber security operations. It is especially relevant in real time systems where accuracy of detection and the speed of response are equally important.

Overall, these AUC values observed for these models particularly the superior performance of MPERGA indicates its effectiveness and reliability in real time malware detection. Its capability to keep high AUC values across different sample sizes shows it's ready for use in numerous and intricate cyber security environments, making it a great boon in guarding digital infrastructures facing developing cyber threats. Similarly, the Specificity levels in figure 8 are observed to be similarly.

We assess the specificity of MPERGA for NTS and it performs strongly in specificity. As an example, such as for a segmental recalled 70k NTS MPERGA's specificity of 84.07% is higher than Malp Miner (77.03%), SPPNet (81.31%), and GCDroid (78.10%). This suggests how efficient MPERGA is in detecting legitimate software and conclude from that, the way to lower the amount of interruptions due to false alarms is.

MPERGA has a high level of specificity for the dataset size, as the dataset size increases. For example, with 170k NTS its specificity is a good 93.83%, better than its competitors. The high specificity implies that

MPERGA should pick up well on malware even as the dataset gets more complicated.

High specificity has significant impact in real time applications. In such sectors as healthcare systems, financial services, industrial control systems, a high specificity rate is crucial, as small errors lead to large business disruptions due to false positives. It makes sure the cyber security measures don't mistakenly paint legitimate, necessary software as suspect and act, resulting in unnecessary, maybe even harmful, disruption.

Additionally, as this scale up process transpires, at 1.2M NTS, MPERGA is able to maintain specificity of 90.81%, indicating that it can properly scale up. This is particularly important in large and complex digital systems with abundant legitimate software and systems, but high cost of false positives.

Additionally, to keep user confidence in cyber security systems, a high specificity rate is a prerequisite. Frequent false positives can also mean your users start to ignore security alerts and stop paying attention to true dangers.

In summary, the observed performance of these models, especially the high performance of MPERGA, suggest that MPERGA is effective in real time malware detection scenarios. MPERGA can achieve high specificity rate at several sample size ranges, indicating the stability and efficiency of MPERGA to differentiate malware from a non-malware. With this in mind, MPERGA is an important tool for anyone who wants to be sure that digital infrastructures are secure and continue to function smoothly, while limiting the disruption resulting from false positives.

5. Conclusion and Future Scopes

This study research has resulted in development and validation of a novel approach — the Malware Prediction Model MPERGA — an innovative approach that combined Auto Encoders and Attention Mechanisms into a Recurrent Graph Relationship Analysis framework. Results of a comprehensive evaluation using Malware Memory Analysis and the Kharon Malware Database show MPERGA to outperform existing models, such as Malp Miner, SPPNet, and GCDroid. For the above-mentioned metrics such as precision, accuracy, recall, delay and AUC, the model has done so in a remarkable fashion.

Existing demographic data shows an 8.3% increase in the ability of MPERGA to do so programmatically over other forms of demographic data and a further 8.5% increase over simple statistical methods. Through a superior recall rate compared to all other classifiers in identifying true instances of malware, the model is clearly effective for preventing system infringements at higher sample sizes. Additionally, AUC values are greater and detection delay is smaller for MPERGA compared to malware detection methods in the literature, highlighting its efficiency and reliability in the classification of malware and non malware instances, and its ability to minimize disruptions introduced by false positives.

MPERGA is indeed having great impacts on cyber security. As an effective way for protecting digital infrastructures, it will be able to adapt to the evolving threats as well as to its ability to handle extensive and complex datasets. Thanks to the high accuracy and fast computational speed, the model can robustly defend against malware attacks, and play a significant role in guarding sensitive systems in financial, medical and national security related sectors.

6. Future Scope

The enhancements and applications of MPERGA are many and look ahead. Future research can focus on the following areas:

- **Integration with IoT and Edge Computing:** With the rapid growth in the number of Internet of Things (IoT) devices, the adaptation of MPERGA targeted for efficient operation in edge computing environments may offer robust security solutions with such immense networks.
- **Adaptation to Zero-Day Malware:** Improving the model's ability to predict zero day attacks, which are one of the biggest problems in cybersecurity, would be a big improvement.
- **Real-Time Application in Diverse Environments:** Moving MPERGA out into the real world with actual corporate networks, but also critical infrastructure, and seeing how it works allows us to learn and further refine the project.
- **Ethical AI Considerations:** The more powerful AI models get, the more crucial will be ensuring that they are being used in an ethical way — i.e. with regards to privacy and data security.
- **Cross-Domain Adaptability:** Other domains, such as fraud detection or intrusion detection systems, could be investigated to see if MPERGA is applicative.

Finally, MPERGA is a giant step forward for cybersecurity. The ability to detect malware in an accurate and efficient manner has huge promise for improving digital security. With future developments and applications of such model, this model can provide not only an advancement of the field of cybersecurity but also enable a safe network which is more and more interconnected and based upon digital technologies.

References

[1] F. Zhong, Z. Chen, M. Xu, G. Zhang, D. Yu, X. Cheng, Malware-on-the-Brain: Illuminating Malware Byte Codes with Images for Malware Classification. IEEE Transactions on Computers, 72(2), (2023) 438-451. https://doi.org/10.1109/TC.2022.3160357

[2] A. bin Asad, R. Mansur, S. Zawad, N. Evan, M.I. Hossain, Analysis of Malware Prediction Based on Infection Rate Using Machine Learning Techniques. IEEE Region 10 Symposium (TENSYMP), IEEE, Bangladesh. https://doi.org/10.1109/TENSYMP50017.2020.9230624

[3] M.F. Abdelwahed, M.M. Kamal, G. Sayed, Detecting Malware Activities with MalpMiner: A Dynamic Analysis Approach. IEEE Access, 11, (2023) 84772-84784. https://doi.org/10.1109/ACCESS.2023.3266562

[4] K. Rana, S. Gupta, G. Kaur, A.L. Yadav, (2024) Malware Detection in Network Traffic using Machine Learning. International Conference on Applied Artificial Intelligence and Computing (ICAAIC), IEEE, India. https://doi.org/10.1109/ICAAIC60222.2024.10575355

[5] O.E. Kural, E. Kiliç, C. Aksaç, Apk2Audio4AndMal: Audio Based Malware Family Detection Framework. IEEE Access, 11, (2023) 27527-27535. https://doi.org/10.1109/ACCESS.2023.3258377

[6] H. Kim, M. Kim, (2024) Malware Detection and Classification System Based on CNN-BiLSTM. Electronics, 13(13), 2539. https://doi.org/10.3390/electronics13132539

[7] K.A. Dhanya, P. Vinod, S.Y. Yerima, A. Bashar, A. David, T. Abhiram, A. Antony, A.K. Shavanas, G. Kumar, Obfuscated Malware Detection in IoT Android Applications Using Markov Images and CNN. IEEE Systems Journal, 17(2), (2023) 2756-2766. https://doi.org/10.1109/JSYST.2023.3238678

[8] Y.H. Chen, S.C. Lin, S.C. Huang, C.L. Lei, C.Y. Huang, Guided Malware Sample Analysis Based on Graph Neural Networks. IEEE Transactions on Information Forensics and Security, 18, (2023) 4128-4143. https://doi.org/10.1109/TIFS.2023.3283913

[9] D.Y.M. Benchadi, B. Batalo, K. Fukui, Efficient Malware Analysis Using Subspace-Based Methods on Representative Image Patterns. IEEE Access, 11, (2023) 102492-102507. https://doi.org/10.1109/ACCESS.2023.3313409

[10] I. Gulatas, H.H. Kilinc, A.H. Zaim, M. A. Aydin, Malware Threat on Edge/Fog Computing Environments From Internet of Things Devices Perspective. IEEE Access, 11, (2023) 33584-33606. https://doi.org/10.1109/ACCESS.2023.3262614

[11] B. Jin, J. Choi, J.B. Hong, H. Kim, On the Effectiveness of Perturbations in Generating Evasive Malware Variants. IEEE Access, 11, (2023) 31062-31074. https://doi.org/10.1109/ACCESS.2023.3262265

[12] J. Jeon, B. Jeong, S. Baek, Y.S. Jeong, Static Multi Feature-Based Malware Detection Using Multi SPP-net in Smart IoT Environments. IEEE Transactions on Information Forensics and Security, 19, (2024) 2487-2500. https://doi.org/10.1109/TIFS.2024.3350379

[13] M. Venkatasubramanian, A.H. Lashkari, S. Hakak, IoT Malware Analysis Using Federated Learning: A Comprehensive Survey. IEEE Access, 11, (2023) 5004-5018. https://doi.org/10.1109/ACCESS.2023.3235389

[14] E.C. Bayazit, O.K. Sahingoz, B. Dogan, Protecting Android Devices from Malware Attacks: A State-of-the-Art Report of Concepts, Modern Learning Models and Challenges. IEEE Access, 11, (2023) 123314-123334. https://doi.org/10.1109/ACCESS.2023.3323396

[15] G.W. Wong, Y.T. Huang, Y.R. Guo, Y. Sun, M.C. Chen, Attention-Based API Locating for Malware Techniques. IEEE Transactions on Information Forensics and Security, 19, (2024) 1199-1212. https://doi.org/10.1109/TIFS.2023.3330337

[16] D.T. Uysal, P.D. Yoo, K. Taha, Data-Driven Malware Detection for 6G Networks: A Survey from the Perspective of Continuous Learning and Explainability via Visualisation. IEEE Open Journal of Vehicular Technology, 4, (2023) 61-71. https://doi.org/10.1109/OJVT.2022.3219898

[17] S. Ali, O. Abusabha, F. Ali, M. Imran, T. Abuhmed, Effective Multitask Deep Learning for IoT Malware Detection and Identification Using Behavioral Traffic Analysis. IEEE Transactions on Network and Service Management, 20(2), (2023) 1199-1209. https://doi.org/10.1109/TNSM.2022.3200741

[18] S. Li, Y. Li, X. Wu, S.A. Otaibi, Z. Tian, Imbalanced Malware Family Classification Using Multimodal Fusion and Weight Self-Learning. in IEEE Transactions on Intelligent Transportation Systems, 24(7), (2023) 7642-7652. https://doi.org/10.1109/TITS.2022.3208891

[19] U. Ahmed, J.C. W. Lin, G. Srivastava, A. Jolfaei, Active Learning Based Adversary Evasion Attacks Defense for Malwares in the Internet of Things. IEEE Systems Journal, 17(2), (2023) 2434-2444. https://doi.org/10.1109/JSYST.2022.3223694

[20] W. Niu, Y. Wang, X. Liu, R. Yan, X. Li, X. Zhang, GCDroid: Android Malware Detection Based on Graph Compression With Reachability Relationship Extraction for IoT Devices. IEEE Internet of Things Journal, 10(13), (2023) 11343-

11356. https://doi.org/10.1109/JIOT.2023.3241697

[21] H. Lee, S. Kim, D. Baek, D. Kim, D. Hwang, Robust IoT Malware Detection and Classification Using Opcode Category Features on Machine Learning. IEEE Access, 11, (2023) 18855-18867. https://doi.org/10.1109/JIOT.2023.3241697

[22] M. Torres, R. Álvarez, M. Cazorla, A Malware Detection Approach Based on Feature Engineering and Behavior Analysis, IEEE Access, 11, (2023) 105355-105367. https://doi.org/10.1109/ACCESS.2023.3319093

[23] L. Huang, J. Xue, Y. Wang, D. Qu, J. Chen, N. Zhang, L. Zhang, EAODroid: Android malware detection based on enhanced API order. Chinese Journal of Electronics, 32(5), (2023) 1169-1178. https://doi.org/10.23919/cje.2021.00.451

[24] H. Manthena, J.C. Kimmel, M. Abdelsalam, M. Gupta, Analyzing and Explaining Black-Box Models for Online Malware Detection. IEEE Access, 11, (2023) 25237-25252. https://doi.org/10.1109/ACCESS.2023.3255176

[25] S. Kasarapu, S. Shukla, S.M. PudukotaiDinakarrao, Resource- and Workload-Aware Model Parallelism-Inspired Novel Malware Detection for IoT Devices. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42(12), (2023) 4618-4628. https://doi.org/10.1109/TCAD.2023.3290128

[26] T.H. Hai, V. Van Thieu, T.T. Duong, H.H. Nguyen, E.N. Huh, A Proposed New Endpoint Detection and Response With Image-Based Malware Detection System. IEEE Access, 11, (2023) 122859-122875. https://doi.org/10.1109/ACCESS.2023.3329112

[27] F. A. Almarshad, M. Zakariah, G. A. Gashgari, E. A. Aldakheel and A. I. A. Alzahrani, Detection of Android Malware Using Machine Learning and Siamese Shot Learning Technique for Security. IEEE Access, 11, (2023) 127697-127714. https://doi.org/10.1109/ACCESS.2023.3331739

[28] T. He, C. Han, R. Isawa, T. Takahashi, S. Kijima, J. Takeuchi, Scalable and Fast Algorithm for Constructing Phylogenetic Trees With Application to IoT Malware Clustering. IEEE Access, 11, (2023) 8240-8253. https://doi.org/10.1109/ACCESS.2023.3238711

[29] Y. He, Y. Liu, L. Wu, Z. Yang, K. Ren, Z. Qin, MsDroid: Identifying Malicious Snippets for Android Malware Detection. IEEE Transactions on Dependable and Secure Computing, 20(3), (2023) 2025-2039. https://doi.org/10.1109/TDSC.2022.3168285

[30] L.d. Costa, V. Moia, A Lightweight and Multi-Stage Approach for Android Malware Detection Using Non-Invasive Machine Learning Techniques. IEEE Access, 11, (2023) 73127-73144. https://doi.org/10.1109/ACCESS.2023.3296606

[31] J. Qiu, Q.L. Han, W. Luo, L. Pan, S. Nepal, J. Zhang, Y. Xiang, Cyber code intelligence for android malware detection. IEEE Transactions on Cybernetics, 53(1), (2022) 617-627. https://doi.org/10.1109/TCYB.2022.3164625

[32] H. Alamro, W. Mtouaa, S. Aljameel, A.S. Salama, M.A. Hamza, A.Y. Othman, Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cyber security. IEEE Access, 11, (2023) 72509-72517. https://doi.org/10.1109/ACCESS.2023.3294263

[33] Y. Zhang, G. Gui, S. Mao, A Lightweight Malware Traffic Classification Method Based on a Broad Learning Architecture, IEEE Internet of Things Journal, 10(23), (2023) 21131-21132. https://doi.org/10.1109/JIOT.2023.3297210

[34] H. Kheddar, Y. Himeur, A.I. Awad, Deep transfer learning for intrusion detection in industrial control networks: A comprehensive review. In Journal of Network and Computer Applications 220, (2023) 103760. https://doi.org/10.1016/j.jnca.2023.103760

[35] A. Gueriani, H. Kheddar, A.C. Mazari, (2023) Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey. International Conference on Electronics, Energy and Measurement (IC2EM), IEEE, Medea. https://doi.org/10.1109/IC2EM59347.2023.10419560

[36] M. Anusha, M. Karthika, Deep Learning Based Maldroid Stacked Propagate Network for Android Malware Prediction for Security Enhancement. Indian Journal of Science and Technology, 17(45), (2024) 4743-4755. https://doi.org/10.17485/IJST/v17i45.3099

[37] C. Duthie, G.J.W. Kathrine, G. Amala Nikitha, S.B. Xavier, I.J. Jebadurai, (2023) Deep Learning based Malware Analysis, Prediction and Prevention, 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), IEEE, India. https://doi.org/10.1109/ICESC57686.2023.10193068

[38] T. Kalpana, (2023) Malware Prediction and Classification for Android Applications Using Machine Learning Techniques. International Conference on Computer Communication and Informatics (ICCCI), IEEE, India. https://doi.org/10.1109/ICCCI56745.2023.10128513

[39] M. Basak, D.W Kim, M.M. Han, G.Y. Shin, Attention-Based Malware Detection Model by Visualizing Latent Features Through Dynamic Residual Kernel Network. Sensors, 24(24), (2024) 7953. https://doi.org/10.3390/s24247953

[40] M. Basak, M.M. Han, CyberSentinel: A Transparent Defense Framework for Malware

Detection in High-Stakes Operational Environments. Sensors, 24(11), (2024) 3406. https://doi.org/10.3390/s24113406

Authors Contribution Statement

Mahesh T. Dhande - Visualization, Methodology, Experimental investigation, Data collection, Formal analysis, Writing-Original draft. Sanjaykumar Tiwari - Conceptualization, Supervision, Coordination of research, Methodology, Validation, Writing - Review of original draft and editing. Nikhil J. Rathod - Supervision, Coordination of research, Validation, Writing - Review of original draft and editing. All the authors read and approved the draft.

Funding

The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Competing Interests

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability

The data supporting the findings of this study can be obtained from the corresponding author upon reasonable request.

Has this article screened for similarity?

Yes

About the License