Yuvaraj Saminathan, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 13, Issue 7, July 2023, pp 56-68

RESEARCH ARTICLE OPEN ACCESS

Time-Series Sequence Generative Adversarial Network for Improving Deep Ensemble Classifier based Ransomware Detection

Yuvaraj Saminathan*, Dr. Robert Lourdusamy
Department of Computer Science, Government Arts College, Coimbatore, Tamilnadu, India.

ABSTRACT
Ransomware is a form of malware that encrypts a user's data and then demands payment to decrypt the data before allowing the user access again. To build an effective defense against such threats, a Deep Ensemble Ransomware Prediction (DeepERPred) model has been designed, it employs Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to spot ransomware and deal with DNA chains that have undergone temporal changes. In contrast, it was time-consuming and costly for manual data labeling, which affects the generation of high-quality training data. This paper therefore creates a data augmentation technique based on a Time-series Sequence Generative Adversarial Network (TSeqGAN) model for creating new digitalized DNA chains from preexisting DNA chains. To solve the GAN discriminating problem, this model uses gradient policy updating, a concept from Reinforcement Learning (RL) that conceptualizes the data generator as a stochastic policy. Using a Monte Carlo (MC) search, the GAN discriminator generates the RL reward necessary for a comprehensive analysis of the complete chain at the intermediate state-action phases. On both natural and artificial chains, it's also use the unsupervised adversarial error in conjunction with a supervised error that is introduced gradually. To learn the causal connections over time and provide useful training sets, this model uses the data's conditional distributions as a stepping stone. Further, the obtained dataset is trained by the ensemble CNN-LSTM classifier to recognize ransomware. Finally, the test outcomes show that the TSeqGAN-DeepERPred model establishes 93.08% accuracy than the existing models for ransomware recognition and categorization.

***Keywords*-**Ransomware, DeepERPred, Time series sequence, GAN, Reinforcement learning, Monte Carlo search, Step-wise supervised error

---

Date of Submission: 24-06-2023 Date of acceptance: 05-07-2023

I. INTRODUCTION

Increased network, operational, and control system deployment coincides with the search engine's ascent as a global user-defined connectivity platform, which in turn has increased the efficiency of social communication and helped to fortify the societies that exist today [1]. Conversely, the digital revolution as a global information system has resulted in a major surge in numerous sorts of cyber assaults. Malware is a type of malicious software that has experienced immense damage to the network platforms as well as criminal acts, deceit, fraud and tribal threat [2-3].

Ransomware is a form of malware that, once installed, secretly encrypts a user's data, rendering it inaccessible, and then demands payment from the victim in exchange for the decryption key [4-6]. Locker ransomware and crypto-ransomware are the two main types of this malicious software. Infiltrating the suspect's network covertly, crypto-ransomware encrypts files on the compromised computer before demanding a ransom [7]. Clients can recover access to their data after making payments and getting the private keys from cybercriminals. When it comes to user data, crypto ransomware frequently targets user-generated records with specific labels like.pdf,.jpg, and.doc files, which often include important information [8]. In the case of locker ransomware, the victim's machine is locked but their data is unharmed. But, access is limited by locking the system resources. It normally locks computing devices or user interfaces and demands ransom for unlocking [9-12].

Because of the rising frequency of ransomware attacks, administrations, businesses, and individuals have been pushed to preserve and backup their sensitive data [13-14]. Yet, due to the incredibly cost-effective nature of these kinds of breaches, the most recent ransomware losses are constantly fluctuating and attackers are continually inventing more complicated software. The current protection strategies for recognizing, assessing and protecting against ransomware are inadequate and incapable of keeping up with the number of assaults. The majority of the strategies are aimed at examining the malware's activities. Developers of ransomware, however, use obfuscation methods such binary code packing to hide their programs from scrutiny.

The term "code packing" refers to a technique used to encrypt both the data and retrieval code within an application [15-16]. While the encapsulated program is executed, the retrieval process program substitutes the original data and restores the activities to their prior configuration. Metamorphic ransomware, which alters its source code in its decrypted pattern [18], produces new layers of malware with each transition, and layered polymorphic malware, which alters its source code and decryption strategy and only reveals a portion of the script at any execution stage. Traditional signature-based behavior analysis anti-malware techniques have a harder time detecting unauthorized code and viral activities when the utility software necessary for a malicious operation is encoded [19]. Such methods of detection, however, have a low success rate.

This prompted Khan et al. [20] to create DNAact-Ran, a machine learning-based ransomware detection technique. Using a digital DNA sequencing model built using machine learning, they came up with a novel approach to detect and classify ransomware. Multi-Objective Grey-Wolf Optimization (MOGWO) and Binary Cuckoo Search (BCS) were first used to extract the primary characteristics from the preprocessed database. After deciding on which features to use, a digital DNA string was constructed for them according to the DNA string model's requirements and the k-mer frequency vector. The DNA code was analyzed, and then goodware or ransomware was determined using Linear Regression (LR). Conversely, this technique did not execute efficiently if the digital DNA strings were modified in time. The LR classifier cannot train satisfactorily to learn temporally modified strings while the digital DNA strings were extremely linear.

To combat these problems, a DeepERPred technique was developed [21], which manages the temporally modified DNA strings to detect ransomware. This newly developed technique integrates CNN and LSTM structures to train the correlation between the different current and previous events of strings properly. According to this training, the chosen characteristics were classified and detected as either ransomware or goodware. But, the manual data labeling was tedious and expensive, which influences the creation of good-quality training data.

Therefore in this manuscript, a Time-series Sequence Generative Adversarial Network (TSeqGAN)-based data augmentation technique is proposed, which gets training sets by creating new digital DNA strings from the available DNA sequences. Since the RL version of this TSeqGAN treats the data generator as a stochastic strategy, it is able to easily avoid the generator differentiation problem. The RL reward signal is generated by a GAN discriminator after a full sequence evaluation, and then it is sent back to the intermediate state-action stages using MC search. Unfortunately, GAN's sequential settings don't take into account the special temporal correlations that come with time-series data. In the TSeqGAN model, the unsupervised adversarial loss is applied to both actual and synthetic sequences, and then a stepwise supervised loss is introduced using the original data as supervision. This model is thus strongly recommended to reflect the gradual conditional distributions present in the data. When the embedding and generator networks are trained together, the supervised loss is minimized, and the latent space serves to improve both parameter efficiency and the generator's ability to understand temporal correlations.

The following subsections make up the rest of this article: Work on detecting ransomware is discussed in Section II. Section III discusses the

proposed methodology, while Section IV demonstrates its efficacy. Section V provides a summary and recommendations for the future.

II. LITERATURE SURVEY

Kim et al. [22] designed a new model for identifying android malware using various characteristics that reflects the behavior of android applications. First, various categories of characteristics were extracted by evaluating the different records. Using such characteristics, an effective feature vector was created and classified by the multimodal deep learning algorithm to identify malware. However, the considered characteristics were static, whereas dynamic characteristics were needed to increase the accuracy.

Al-rimy et al. [23] developed incremental Bagging (iBagging) and Enhanced Semi-Random Subspace (ESRS) selection to produce an ensemble-based identification framework. Initially, iBagging was used to create successive subsets in a manner that is consistent with the stages of assault experienced by crypto-ransomware. In order to give the most advantageous, noise-free, and varied feature subspaces possible, a group of classifiers was trained with ESRS. In the end, a grid search was used to choose the best ensemble of basic classifiers, and ransomware was recognized by a simple majority. Selecting redundant characteristics in each subspace independently from the others reduced accuracy.

Bibi et al. [24] developed an effective deep learning-based malware identification framework for competent and enhanced ransomware recognition in the Android platform by using the LSTM. First, various features were extracted and chosen based on the majority voting. Then, the selected features were classified by the LSTM to identify malware. But, it needs a more advanced ensemble model to increase the accuracy.

For discovering ransomware, Khammas [25] created an innovative approach based on the static evaluation. Initially, functional information was standardized to extract the characteristics and common themes connected with the relevant entries. After applying the gain ratio to get rid of the duplicate features, the most important features were assigned to goodware or ransomware using the Random Forest (RF) method. However, the efficiency was influenced if the number of instances was high.

A non-signature based identification method based on efficient windows API call sequences was developed by Ahmed et al. [26]. Therefore, it uses an Enhanced Maximum-Relevance and Minimum-Redundancy (EmRmR) filter to determine which traits best characterize the ransomware and exclude the rest. However, it was time-consuming as the number of characteristics increased.

To find the unsupervised, underlying origins of the novel variants' divergent patterns, Sharmeen et al. [27] constructed a semi-supervised model utilizing deep learning techniques. This model was able to scale to incorporate new malicious executables since it identified inherent properties in a variety of patterns from unlabeled ransomware collected in the wild. Then combined the unsupervised model with a supervised classifier to create an adaptive ransomware detection system. But, it needs an ensemble model to further improve the detection accuracy.

Catak et al. [28] suggested an image augmentation enhanced CNN structures to identify malware classes in a metamorphic malware scenario. Initially, the malware data was acquired and transformed into RGB images by the windowing method. Then, those images were augmented and classified by the different CNN structures to identify malware classes. But, the efficiency was not satisfactory because of using RGB training image sets.

Malware classification systems, such DFS-MC and DBFS-MC from Asam et al. [29], are based on deep feature spaces. The DFS-MC used an SVM to classify malware based on deep properties derived from the modified CNN structures. By combining the deep feature spaces of the redesigned CNN structures, the discrimination ability was enhanced in the DBFS-MC. Then, the DBFS was provided to the SVM for identifying the exceptional malware. But, it must be adapted to identify new malware threats.

To extract different properties from crypto-ransomware at the dynamic link library, function call, and assembly levels, Poudyal and Dasgupta [30] developed an AI-enabled Hybrid Method (AIHM). Improved static and dynamic methodologies, as well as a novel system for evaluating behavioral chains using AI methods, are all part of this hybrid model. Additionally, ransomware was detected using a combination of association rule mining and machine learning classifiers. But, the number of training samples was limited and few samples were not effective for dynamic analysis.

III. PROPOSED METHODOLOGY

In this part, the TSeqGAN with DeepERPred model is described in brief. An entire pipeline of this study is illustrated in Fig 1.

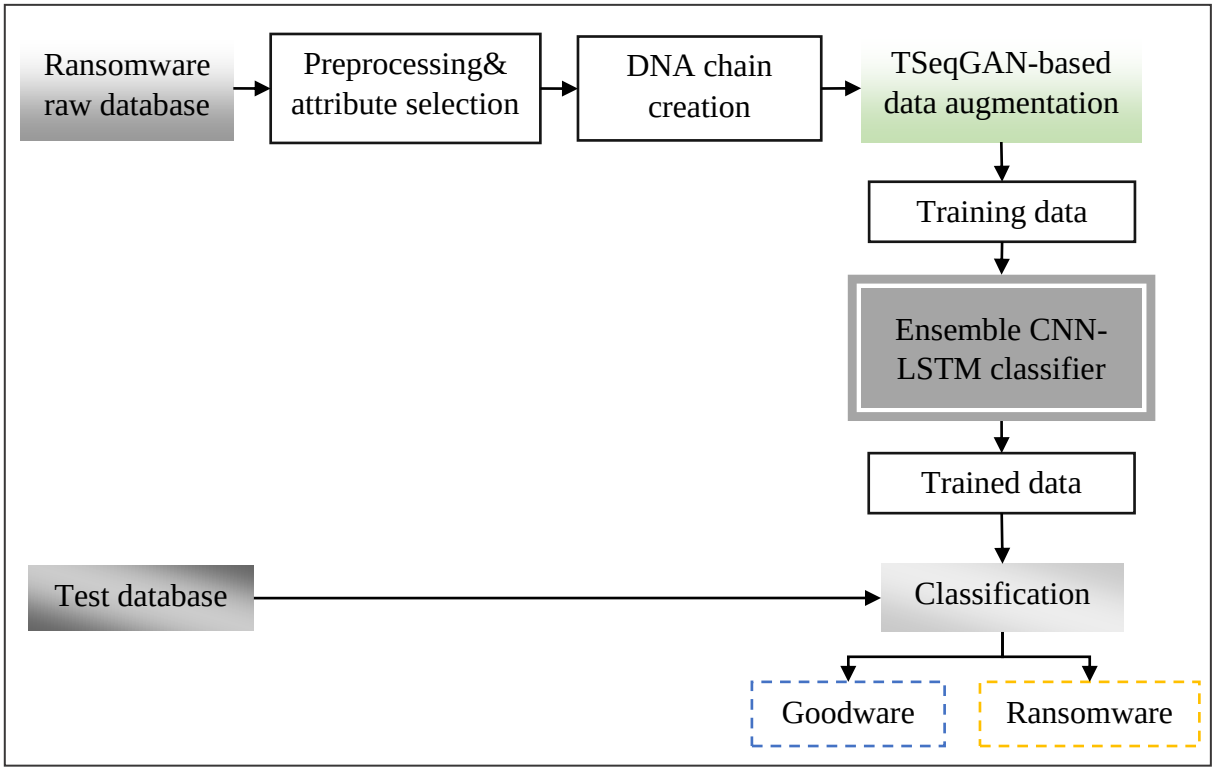

Figure 1. Overall Pipeline of Ransomware Recognition Model

3.1 DNA Chain Generation

The method begins with the collection and preprocessing of a raw ransomware database in order to convert the records into the necessary format. So, the lost, improper and noisy records are discarded from the real database. Afterward, highly significant characteristics are decided by the MOGWO and BCS algorithms to minimize the attribute dimensionality [12]. Once all the attributes are chosen, the digitalized DNA chain is produced by calculating design constraints and k-mer frequency map. This digitalized DNA chain makes the training set, which is further augmented by the TSeqGAN model.

3.2 Time-series Sequence Generative Adversarial Network

There are 4 main parts of TSeqGAN: sequence generator, sequence discriminator, and embedding and regeneration functions. TSeqGAN's main objective is to train the auto-encoder units alongside the adversarial units in order to encode attributes, create interpretations, and iterate through intervals simultaneously. The latent space in which the adversarial network functions is provided by the embedding network. Furthermore, the latent dynamics of both natural and artificial records are matched using a guided loss.

3.2.1 Embedding and Regeneration Functions

Through low-dimensional interpretations enabled by the embedding and regeneration functions, the adversarial network can understand the underlying temporal dynamics of the records. Latent vector spaces $\mathcal{H}_S$ and $\mathcal{H}_X$ are associated with feature spaces S and X, respectively. Next, to extract static and temporal features of their latent codes $h_S, h_{1:T} = e(s, x_{1:T})$, thanks to the embedding function $e: S \times \prod_t X \to \mathcal{H}_S \times \prod_t \mathcal{H}_X$. The recurrent network implements this e as:

$$h_S = e_S(s),\ h_t = e_X(h_S, h_{t-1}, x_t) \qquad (1)$$

Embedding networks for static features $e_S: S \to \mathcal{H}_S$ and temporal characteristics $e_X: \mathcal{H}_S \times \mathcal{H}_X \times X \to \mathcal{H}_X$ are both represented by the equation (1).

To get back from static and temporal codes to their feature interpretations $\tilde{s}, \tilde{x}_{1:T} = r(h_S, h_{1:T})$, it employs the regeneration function $r: \mathcal{H}_S \times \prod_t \mathcal{H}_X \to S \times \prod_t X$. This r is executed by the feed-forward network at all iterations as:

$$\tilde{s} = r_S(h_s),\ \tilde{x}_t = r_X(h_t) \qquad (2)$$

Networks for both static and dynamic embedding regeneration are denoted by $r_S: \mathcal{H}_S \to S$ and $r_X: \mathcal{H}_X \to X$, respectively, in Eq. (2). Note that any selection structure can be used to determine the parameters for these functions; the only requirements are that they are auto-regressive and follow causal ordering (i.e., the results at any iteration solely depend on the data from previous iterations). In order to identify the origin of the gains in this analysis, the developers perform Eqns. (1) and (2) as minimal examples.

3.2.2 Sequence Adversarial Network

The first output of the generator is in the embedding space, rather than in the feature space, where synthetic results can be created. For a database of real-time structured DNA chains, θ-parameterized generative model G_θ is trained to create a chain $Y_{1:T} = (y_1, \ldots, y_t, \ldots, y_T), y_t \in \mathcal{Y}$, where $\mathcal{Y}$ represents all possible tokens. On the basis of the RL, it is assumed that. The set of tokens that have been created up until this point $(y_1, \ldots, y_{t-1})$ constitutes the state $\mathcal{s}$ of the system, and the next token to be selected is the action a. After an action is chosen, the transition between states is predetermined in the stochastic policy model $G_\theta(y_t | Y_{1:t-1})$, where $\delta^a_{\mathcal{s},\mathcal{s}'} = 1$ for the next state $\mathcal{s}' = Y_{1:t}$ when $\mathcal{s} = Y_{1:t-1}$ and $a = y_t$ and $\mathcal{s}'', \delta^a_{\mathcal{s},\mathcal{s}''} = 0$. for other next states.

To further improve G_θ, and also train a φ-parameterized discriminative model D_φ. $D_\varphi(Y_{1:T})$ is a probability that indicates whether or not the chain $Y_{1:T}$ is likely to have been derived from the actual DNA chain information.

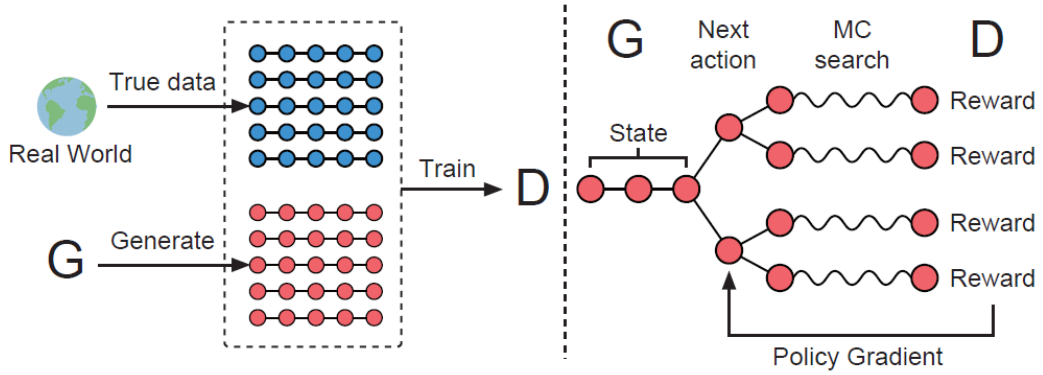

Figure 2. Structure of TSeqGAN. Left: D is trained over the actual data and the created data by G. Right: G is trained by policy gradient, where the final reward signal is provided by D and is fed back to the intermediate action value by MC search

Fig 2 shows how this D_φ is trained using real DNA chain data as positive examples and synthetic DNA chains generated by G_θ as negative examples. Simultaneously, the expected and reward from D_φ are used to fine-tune $G_\theta$via a policy gradient and MC search. The incentive is proportional to the likelihood that it would fool D_φ.

TSEQGAN VIA POLICY GRADIENT

From a seed state s_0, the generative network (policy) $G_\theta(y_t|Y_{1:t-1})$ will produce a DNA chain with the goal of optimizing the predicted reward at the end of the process.

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1 \in y} G_\theta(y_1|s_0) \cdot Q_{D_\varphi}^{G_\theta}(s_0, y_1) \quad (3)$$

A full DNA chain is rewarded as shown by R_T in Eq. (3). Take note that the incentive comes from D_φ, $Q_{D_\varphi}^{G_\theta}(s, a)$ meaning the expected cumulative reward after starting with s, taking on a, and adhering to policy G_θ in the DNA chain.

Given an initial state, the goal of the generator is to produce the DNA chain that would convince the discriminator that the fitness function is correct. Moreover, the reinforce strategy is utilized and the predicted probability of being true is considered by the discriminator $D_\varphi(Y_{1:T}^n)$ as the reward. So,

$$Q_{D_\varphi}^{G_\theta}(a| = y_T, s = Y_{1:T-1}) = D_\varphi(Y_{1:T}) \quad (4)$$

The discriminator, however, only provides incentive for a full chain. Because of bothering about the long-term reward, at all intervals, it needs to think about the ultimate outcome in addition to the goals of preceding tokens (prefix). Using a subset of the unknown final $T - t$ tokens, the MC search is combined with the roll-out strategy G_β to investigate the action-value for a transitional stage. The formal definition of an N -time MC search is

$$\{Y_{1:T}^1, \dots, Y_{1:T}^N\} = MC^{G_\beta}(Y_{1:t}; N) \quad (5)$$

In Eq. (5), $Y_{1:t}^n = (y_1, \dots, y_t)$ and $Y_{t+1:T}^n$ is sampled depending on G_β (which is assigned similar as the generator) and the present state. N iterations of the roll-out policy are executed from the current state to the end of the chain to decrease variation and ensure an accurate assessment of the action's worth.

$$Q_{D_\varphi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\varphi(Y_{1:T}^N), Y_{1:T}^N \in MC^{G_\beta}(Y_{1:t}; N), & for\ t < T \\ D_\varphi(Y_{1:t}), & for\ t = T \end{cases} \quad (6)$$

In Eq. (6), if there is no reward at the intermediate stage, the function is represented iteratively as the next-state value, starting at state $s' = Y_{1:t}$ and continuing until the end.

D_φ can be used as a reward function because it will be adaptively adjusted to improve D_φ in a recursive fashion. After obtaining a collection of more realistic created chains, the discriminator model is retrained as:

$$\min_\varphi -\mathbb{E}_{Y \sim p_{data}}[\log D_\varphi(Y)] - \mathbb{E}_{Y \sim G_\theta}\left[log\left(1 - D_\varphi(Y)\right)\right] \quad (7)$$

The generator must be modified after every time the new discriminator model is constructed. In order to maximize the long-term payoff, this policy-based approach must optimize a parameterized policy. Then, calculate the fitness function's gradient $J(\theta)$, relative to the generator's parameters θ, by

$$\nabla_\theta J(\theta) = \mathbb{E}_{Y_{1:t-1} \sim G_\theta}\left[\sum_{y_t \in y} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\varphi}^{G_\theta}(Y_{1:t-1}, y_t)\right] \quad (8)$$

The above term is because of the deterministic state transition and zero intermediary rewards. Utilizing probability fractions, an unbiased analysis on a single episode is made for Eq. (8) as:

$$\nabla_\theta J(\theta) \simeq \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t \in y} \nabla_\theta G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\varphi}^{G_\theta}(Y_{1:t-1}, y_t) \quad (9)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t \in y} G_\theta(y_t|Y_{1:t-1}) \nabla_\theta \log G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\varphi}^{G_\theta}(Y_{1:t-1}, y_t)$$

$$= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta(y_t|Y_{1:t-1})}\left[\nabla_\theta \, log \, G_\theta(y_t|Y_{1:t-1}) \cdot Q_{D_\varphi}^{G_\theta}(Y_{1:t-1}, y_t)\right]$$

The observed intermediate state sampled from G_θ is denoted here by $Y_{1:t-1}$. Variables in the generator are adjusted as follows because of the approximation of the expectation $\mathbb{E}[\cdot]$ using sampling methods:

$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta) \quad (10)$$

In Eq. (10), $\alpha_h \in \mathbb{R}^+$ is the related training rate at h^{th} iteration.

Generator for DNA Chains

As the basis for this generative model, they employ a Recurrent Neural Network (RNN). The RNN, using the recursive update function g, maps the input embedding interpretations $x_1, \dots, x_T$ of the DNA chain $x_1, \dots, x_T$ into the chain of hidden states $h_1, \dots, h_T$.

$$h_t = g(h_{t-1}, x_t) \quad (11)$$

Additionally, the output token distribution is mapped from the hidden states using a softmax layer (z) as:

$$p(y_t|x_1, \dots, x_t) = z(h_t) = softmax(c + Vh_t) \quad (12)$$

V is a weight matrix and c is a bias vector in Eq. (12). The LSTM cells use g in Eq. (11) to overcome the backpropagation via interval problem of vanishing and exploding gradients. It is obvious that the Gated Recurrent Unit (GRU) and soft attention strategy are employed as a generator in TSeqGAN.

Discriminator for DNA Chains

The CNN is adopted as the discriminator because of its efficiency. In this study, the discriminator estimates the probability that a complete chain is true. Initially, an input chain $x_1, \dots, x_T$ is defined as:

$$\mathcal{E}_{1:T} = x_1 \oplus x_2 \oplus \dots \oplus x_T \quad (13)$$

To produce the matrix $\mathcal{E}_{1:T} \in \mathbb{R}^{T \times k}$, the k-dimensional token embedding is denoted by $x_t \in \mathbb{R}^k$ in Eq. (13), and $\oplus$ denotes the concatenation operator. Then, a unique feature map is generated by applying a convolutional process with a window size of l words using a kernel $w \in \mathbb{R}^{l \times k}$, as:

$$c_i = \rho(w \otimes \mathcal{E}_{i:i+l-1} + b) \quad (14)$$

In Eq. (14), $\otimes$ refers to the summation of element-wise production, b denotes the bias term and ρ denotes the non-linear function. Different number of kernels is utilized with various window sizes to capture multiple characteristics. In addition, the feature maps are subjected to a max-over-interval pooling procedure, written as $\tilde{c} = \max\{c_1, \dots, c_{T-l+1}\}$.

In order to maximize the usefulness of the combined feature maps, a highway-like framework has also been integrated. The likelihood that the input chain is correct is then determined using a sigmoid-activated fully-connected layer. The goal of optimization is to find the value of x that minimizes the derivative of the supervised error between the ground-truth class and the expected probability as given by Eq. (7).

3.2.3 Mutual Training

The embedding and regeneration functions, as a mapping between feature and latent spaces, must provide precise restorations $\tilde{s}, \tilde{x}_{1:T}$ of the actual record $s, x_{1:T}$ from their latent interpretations $h_S, h_{1:T}$. So, the primary fitness function is the restoration loss as:

$$\mathcal{L}_R = \mathbb{E}_{s,x_{1:T} \sim p}[\|s - \tilde{s}\|_2 + \textstyle\sum_t \|x_t - \tilde{x}_t\|_2] \quad (15)$$

In this TSeqGAN, the generator considers synthetic embeddings $\hat{h}_S, \hat{h}_{1:t-1}$ to create the next synthetic vector $\hat{h}_t$ in open-loop mode. Following, the gradients are determined on the unsupervised loss. This helps to enable increasing and reducing the probability of obtaining proper classifications $\hat{y}_S, \hat{y}_{1:T}$ for both the training data $h_S, h_{1:T}$ and for synthetic result $\hat{h}_S, \hat{h}_{1:T}$ from the generator as.

$$\mathcal{L}_U = \mathbb{E}_{s,x_{1:T} \sim p}[\log y_S + \textstyle\sum_t log \, y_t] + \mathbb{E}_{s,x_{1:T} \sim p}[log(1 - \hat{y}_S) + \textstyle\sum_t log(1 - \hat{y}_t)] \quad (16)$$

To increase the efficiency, the generator is trained in closed-loop mode, where it obtains chains of embeddings of real data $h_{1:t-1}$ to create the next latent vector. A loss function is used to calculate the gradient, which is the difference between the two distributions $p(H_t|H_S, H_{1:t-1})$ and $\hat{p}(H_t|H_S, H_{1:t-1})$. So, the supervised loss is obtained by the maximum probability as:

$$\mathcal{L}_S = \mathbb{E}_{s,x_{1:T}\sim p}[\sum_t \|h_t - G_\theta(h_S, h_{t-1}, z_t)\|_2] \quad (17)$$

In Eq. (17), $G_\theta(h_S, h_{t-1}, z_t)$ approximates $\mathbb{E}_{z_t\sim N}[\hat{p}(H_t|H_S, H_{1:t-1}, z_t)]$ with single data z_t. In conclusion, during each training iteration, the DNA chain calculates the difference between the natural and artificial next-step latent vectors. When $\mathcal{L}_U$ pushes the generator to produce realistic chains, further $\mathcal{L}_S$ guarantees that it creates analogous stepwise transitions.

Optimization: Consider $\theta_e, \theta_r, \theta_g$ and θ_d are variables of embedding, regeneration, generator and discriminator units. The first two units are trained using supervised loss and restoration data as:

$$\min_{\theta_e,\theta_r}(\lambda\mathcal{L}_S + \mathcal{L}_R) \quad (18)$$

Two losses are kept in check by a single hyperparameter denoted by $\lambda \geq 0$ in Eq. (18). With $\mathcal{L}_S$, the embedding objective is broadened beyond its original intent of minimizing the size of the adversarial training space. The generator has been trained to look for patterns in time. When training the generator and discriminator units, an adversarial setting is used because:

$$\min_{\theta_g}\left(\eta\mathcal{L}_S + \max_{\theta_d}\mathcal{L}_U\right) \quad (19)$$

In Eq. (19), $\eta \geq 0$ denotes the other hyperparameter that balances 2 losses. The generator reduces the supervised loss and maximizes the efficiency. Training TSeqGAN in this manner allows it to encode (feature vectors), generate (latent interpretations), and iterate (across interval) all at once.

3.3 Deep Ensemble Ransomware Prediction Model

After generating the synthetic DNA chains, the training set is obtained including with the actual DNA chains. The ensemble CNN-LSTM classifier is then used to determine if the input data is ransomware or goodware based on the training sets.

Algorithm:

Input: Raw ransomware database

Output: Ransomware or goodware

Begin

Cleaning up the raw ransomware database of duplicates and other unwanted data;

Determine the most important features by applying the MOGWO and BCS algorithms;

Find the k-mer frequency map and design constraints to make digital DNA chains for training;

Perform TSeqGAN to produce the high-quality training databases with the help of real and synthetic DNA chains;

Train the ensemble CNN-LSTM classifier and validate it using the test data;

Recognize whether the given data is ransomware or goodware;

If ransomware data is recognized, classify their labels;

End

IV. EXPERIMENTAL RESULTS

Here, the TSeqGAN-DeepERPred model is run in JAVA 1.8 to evaluate its efficacy. Its efficiency is also measured against that of currently used models. DNAact-Ran [20], DeepERPred [21], DBFS-MC [29] and AIHM [30]. This analysis considers a real-world corpus available in https://github.com/PSJoshi/Notes/wiki/databases. Overall, there are 30970 attributes among 1524 instances (582 ransomwares and 942 goodware). The term "ransomware" encompasses a wide variety of malicious programs, some of which include

Critroni, CryptLocker, CryptoWall, KOLLAH, Kovter, Locker, MATSNU, PGPCODER, Reveton, TeslaCrypt, and Trojan-Ransom.

Instances of APIs (API), changes to missing files (DROP), operations on registry codes (REG), file operations (FILES), Document Index operations (DIR), embedded chains (STR) and a change to files occupied in file operations (FILES_EXT), are all taken into account.

4.1 Accuracy

It determines how many correct identifications there were out of all tested occurrences.

$$Accuracy = \frac{True\ Positive\ (TP)+True\ Negative\ (TN)}{TP+TN+False\ Positive\ (FP)+False\ Negative\ (FN)} \quad (20)$$

When TP appears in Eq. (20), the classifier considers the ransomware and itself to be identical; when TN appears, the classifier considers the goodware and itself to be identical; when FP appears, the classifier considers the ransomware and the goodware to be indistinguishable; and when FN appears, the classifier considers the goodware and the ransomware to be indistinguishable.

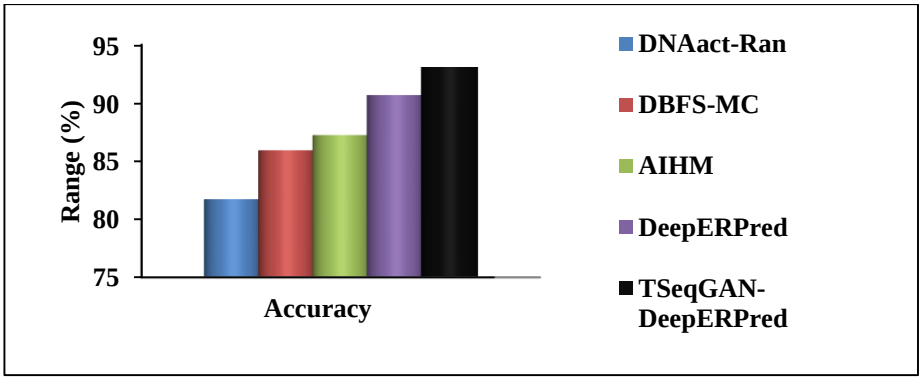

Figure 3. Comparison of Accuracy

The various ransomware recognition and classification models' accuracy (in %) is shown in Fig3. By improving the training samples and the ransomware classification, the TSeqGAN-DeepERPred model outperforms the DNAact-Ran, DBFS-MC, AIHM, and DeepERPred models by a margin of 13.89%, 8.31%, 6.68%, and 2.66%, respectively. The TSeqGAN-DeepERPred model improves the accuracy of identifying and classifying ransomware labels.

4.2 Precision

It calculates the number of ransomware cases that are exactly recognized at both TP and FP.

$$Precision = \frac{TP}{TP+FP} \quad (21)$$

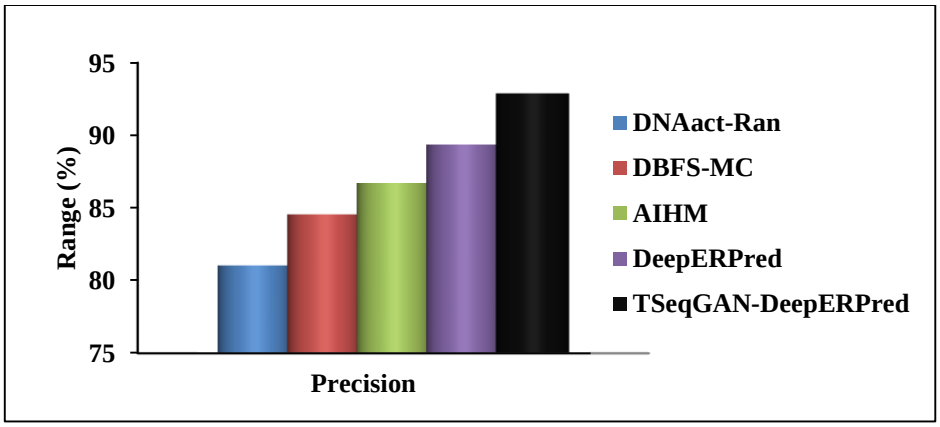

Figure 4. Comparison of Precision

Figure 4 shows the precision (in %) of many ransomware detection and classification models. By automatically labeling ransomware data with the aid of the TSeqGAN structure, it is discovered that the TSeqGAN-DeepERPred model has a precision that is 14.61% higher than the DNAact-Ran model, 9.85% higher than the DBFS-MC model, 7.12% higher than the AIHM model, and 3.94% higher than the DeepERPred model. This demonstrates that TSeqGAN-DeepERPred is superior to existing ransomware recognition classification methods in terms of precision.

4.3 Recall

At both TP and FN, it calculates the fraction of ransomware samples that can be correctly identified.

$$Recall = \frac{TP}{TP+FN} \quad (22)$$

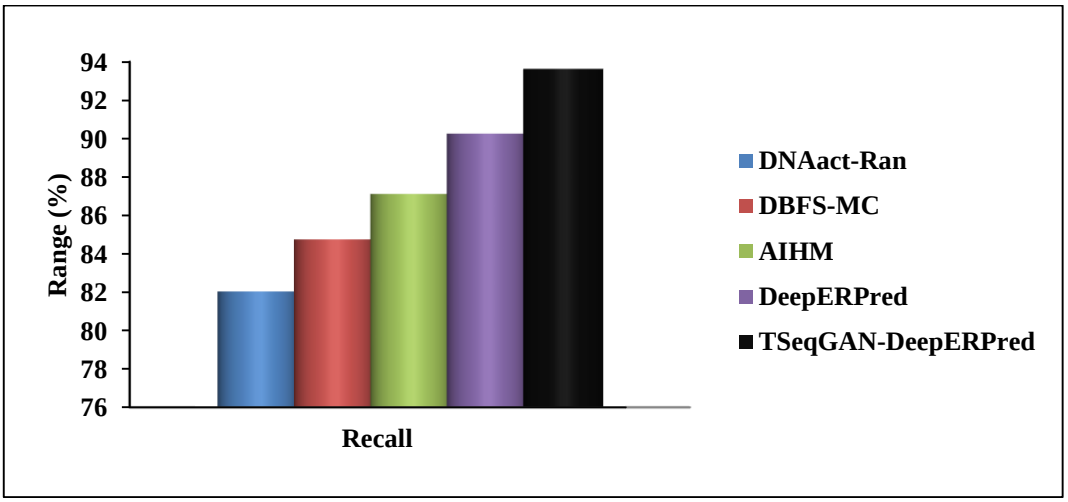

Figure5. Comparison of Recall

Fig 5 displays the recall (in %) for many ransomware recognition and classification methods. Recall for TSeqGAN-DeepERPred is shown to be 14.13% greater than that of DNAact-Ran models, 10.48% higher than DBFS-MC models, 7.47% higher than AIHM models, and 3.7% higher than DeepERPred models. TSeqGAN-DeepERPred's recall is higher than that of any competing ransomware classification model, as this demonstrates.

4.4 F-measure

Harmonic mean recall and precision.

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (23)$$

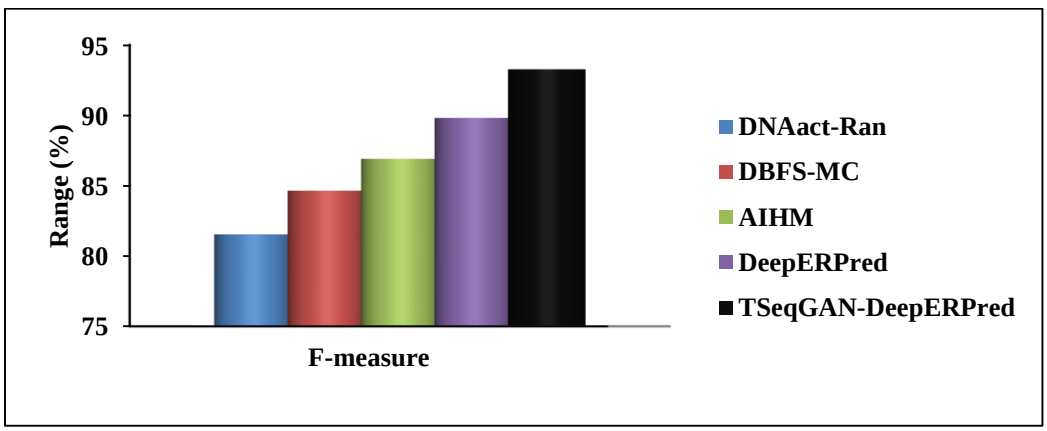

Figure 6. Comparison of F-measure

The f-measure (in %) of several ransomware identification and classification models is shown in Fig 6. Effective data labeling utilizing the TSeqGAN structure is seen to increase the recall of the TSeqGAN-DeepERPred models by 14.36% over the DNAact-Ran, 10.16% over the DBFS-MC, 7.29% over the AIHM, and 3.83% over theDeepERPred models. All other ransomware recognition classification methods are shown to have a lower f-measure than TSeqGAN-DeepERPred.

4.5 Error Rate

It takes into account the total number of tests and determines what percentage of occurrences have faulty recognition.

$$Error\ rate = \frac{FP+FN}{TP+TN+FP+FN} \qquad (24)$$

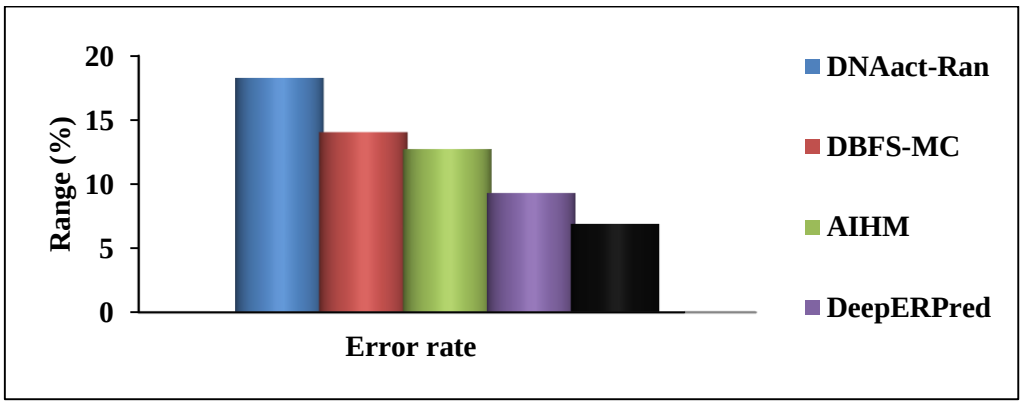

Figure7. Comparison of Error Rate

Error rates (in %) across various ransomware recognition and classification methods are shown in Fig 7. The results show that compared to the DNAact-Ran model, the TSeqGAN-DeepERPred model has a 62.12% lower error rate, the DBFS-MC model has a 50.78% lower error rate, the AIHM model has a 45.73% lower error rate, and the DeepERPred model has a 25.83% lower error rate. In conclusion, the TSeqGAN-DeepERPred model outperforms all others in terms of reducing the error rate during ransomware recognition and classification.

V. CONCLUSION

In this article, the TSeqGAN model was developed to produce new digitalized DNA chains from the available ones for ransomware recognition. Primarily, the ransomware database was transformed into the desired format. Following, MOGWO and BCS were employed to decide the most relevant characteristics to construct the training set, which was further provided to the TSeqGAN to create high-quality training sets by modeling the temporal correlation using the synthetic ransomware data. Additionally, the ransomware labels were classified using an ensemble CNN-LSTM classifier fed with the aforementioned training sets. The experimental findings showed that the TSeqGAN-DeepERPred model had a 93.08% accuracy and an error rate of 6.92% when used for ransomware recognition and classification.

REFERENCES

[1]. B. Vignau, R.Khoury, S.Halléand A.Hamou-Lhadj, The evolution of IoT malwares, from 2008 to 2019: survey, taxonomy, process simulator and perspectives. Journal of Systems Architecture, 116, 2021, 1-32.

[2]. M. N. Alenezi, H.Alabdulrazzaq, A. A. Alshaher and M. M.Alkharang, Evolution of malware threats and techniques: a review. International Journal of Communication Networks and Information Security, 12(3), 2020, 326-337.

[3]. P. Maniriho, A. N. Mahmoodand M. J. M. Chowdhury,A study on malicious software behaviour analysis and detection techniques: taxonomy, current trends and challenges. Future Generation Computer Systems, 130, 2021, 1-18.

[4]. S. L. Popoola, S. O. Ojewande, F. O. Sweetwilliams, S. N. John, and A. A.Atayero, Ransomware: current trend, challenges, and

research directions. In Proceedings of the World Congress on Engineering and Computer Science, I, 2017, pp. 1-6.

[5]. N. Shahand M.Farik, Ransomware-threats vulnerabilities and recommendations. International Journal of Scientific & Technology Research, 6(06), 2017, 307-309.

[6]. B. A. S. Al-rimy, M. A. Maarof and S. Z.M. Shaid, Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions. Computers &Security, 74, 2018, 144-166.

[7]. I. Yaqoob, E. Ahmed, M. H.urRehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran and M.Guizani, The rise of ransomware and emerging security challenges in the Internet of Things. Computer Networks, 129, 2017, 444-458.

[8]. S. Aurangzeb, M.Aleem, M. A. Iqbaland M. A. Islam, Ransomware: a survey and trends. Journal of Information Assurance & Security, 6(2), 217, 48-58.

[9]. V. R. Paşcaand E.Simion, Challenges in cyber security: ransomware phenomenon. In Cyber-Physical Systems Security, Springer, Cham, 2018, pp. 303-330.

[10]. I. Nadir and T.Bakhshi, Contemporary cybercrime: a taxonomy of ransomware threats & mitigation techniques. In IEEE International Conference on Computing, Mathematics and Engineering Technologies,pp. 1-7, 2018.

[11]. M. Humayun, N. Z. Jhanjhi, A.Alsayatand V. Ponnusamy, Internet of things and ransomware: evolution, mitigation and prevention. Egyptian Informatics Journal, 22(1), 2021, 105-117.

[12]. A. Kapoor, A. Gupta, R. Gupta, S.Tanwar, G. Sharma and I. E. Davidson, Ransomware detection, avoidance, and mitigation scheme: a review and future directions. Sustainability, 14(1), 2022, 1-24.

[13]. F. Malecki, Best practices for preventing and recovering from a ransomware attack. Computer Fraud & Security, 2019(3), 8-10.

[14]. T. R. Reshmi, Information security breaches due to ransomware attacks-a systematic literature review. International Journal of Information Management Data Insights, 1(2), 2021, 1-10.

[15]. D. Gibert, C. MateuandJ. Planes, The rise of machine learning for detection and classification of malware: research developments, trends and challenges. Journal of Network and Computer Applications, 153, 2020, 1-22.

[16]. C. Beaman, A.Barkworth, T. D.Akande, S.Hakakand M. K. Khan, Ransomware: recent advances, analysis, challenges and future research directions. Computers & Security, 111, 2021, 1-22.

[17]. H. Xu, Y. Zhou, J. Ming and M.Lyu, Layered obfuscation: a taxonomy of software obfuscation techniques for layered security. Cybersecurity, 3(1), 2020, 1-18.

[18]. N. K. Popliand A. Girdhar, Behavioural analysis of recent ransomwares and prediction of future attacks by polymorphic and metamorphic ransomware. In Computational Intelligence: Theories, Applications and Future Directions-Volume II, Springer, Singapore, 2019, pp. 65-80.

[19]. S. K. Sahay, A. Sharmaand H.Rathore, Evolution of malware and its detection techniques. In Information and Communication Technology for Sustainable Development, Springer, Singapore, 2020, pp. 139-150.

[20]. F. Khan, C.Ncube, L. K.Ramasamy, S. Kadryand Y. Nam, A digital DNA sequencing engine for ransomware detection using machine learning. IEEE Access, 8, 2020, 119710-119719.

[21]. S.Yuvaraj,L.Robert, Deep ensemble classifier for ransomware identification using digitalized DNA genotyping system,international Journal of Intelligent Engineering & Systems,15(16),2022,503-510.

[22]. T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, A multimodal deep learning method for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 2018, 773-788.

[23]. B. A. S. Al-rimy, M. A. Maarofand S. Z. M. Shaid, Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. Future Generation Computer Systems, 101, 2019, 476-491.

[24]. I. Bibi, A.Akhunzada, J. Malik, G. Ahmedand M. Raza, An effective Android ransomware detection through multi-factor feature filtration and recurrent neural network. In IEEE UK/China Emerging Technologies, 2019, pp. 1-4.

[25]. B. M. Khammas, Ransomware detection using random forest technique. ICT Express, 6(4), 2020, 325-331.

[26]. Y. A. Ahmed, B.Koçer, S. Huda, B. A. S. Al-rimyand M. M. Hassan, A system call refinement-based enhanced minimum

redundancy maximum relevance method for ransomware early detection. Journal of Network and Computer Applications, 167, 2020, 1-18.

[27]. S. Sharmeen, Y. A. Ahmed, S. Huda, B. Ş. Koçerand M. M. Hassan, Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches. IEEE Access, 8, 2020, 24522-24534.

[28]. F. O. Catak, J. Ahmed, K.Sahinbasand Z. H.Khand, Data augmentation based malware detection using convolutional neural networks. PeerJ Computer Science, 7, 2021, 1-26.

[29]. M. Asam, S. J. Hussain, M. Mohatram, S. H. Khan, T. Jamal, A. Zafarand U.Zahoora, Detection of exceptional malware variants using deep boosted feature spaces and machine learning. Applied Sciences, 11(21), 2021, 1-16.

[30]. S. Poudyaland D.Dasgupta, Analysis of crypto-ransomware using ML-based multi-level profiling. IEEE Access, 9, 2021, 122532-122547.