future internet

MDPI

Article

An Enhanced Minimax Loss Function Technique in Generative Adversarial Network for Ransomware Behavior Prediction

Mazen Gazzan [1,2] and Frederick T. Sheldon [1,*]

[1] Department of Computer Science, College of Engineering, University of Idaho, Moscow, ID 83844, USA; gazz6545@vandals.uidaho.edu or mzgazzan@nu.edu.sa
[2] College of Computer Science and Information Systems, Najran University, Najran P.O. Box 1988, Saudi Arabia
* Correspondence: sheldon@ieee.org or sheldon@uidaho.edu; Tel.: +1-208-292-2545

Abstract: Recent ransomware attacks threaten not only personal files but also critical infrastructure like smart grids, necessitating early detection before encryption occurs. Current methods, reliant on pre-encryption data, suffer from insufficient and rapidly outdated attack patterns, despite efforts to focus on select features. Such an approach assumes that the same features remain unchanged. This approach proves ineffective due to the polymorphic and metamorphic characteristics of ransomware, which generate unique attack patterns for each new target, particularly in the pre-encryption phase where evasiveness is prioritized. As a result, the selected features quickly become obsolete. Therefore, this study proposes an enhanced Bi-Gradual Minimax (BGM) loss function for the Generative Adversarial Network (GAN) Algorithm that compensates for the attack patterns insufficiency to represents the polymorphic behavior at the earlier phases of the ransomware lifecycle. Unlike existing GAN-based models, the BGM-GAN gradually minimizes the maximum loss of the generator and discriminator in the network. This allows the generator to create artificial patterns that resemble the pre-encryption data distribution. The generator is used to craft evasive adversarial patterns and add them to the original data. Then, the generator and discriminator compete to optimize their weights during the training phase such that the generator produces realistic attack patterns, while the discriminator endeavors to distinguish between the real and crafted patterns. The experimental results show that the proposed BGM-GAN reached maximum accuracy of 0.98, recall (0.96), and a minimum false positive rate (0.14) which all outperform those obtained by the existing works. The application of BGM-GAN can be extended to early detect malware and other types of attacks.

Keywords: ransomware; Generative Adversarial Network; minimax loss function; ransomware detection and prediction; deep learning

Citation: Gazzan, M.; Sheldon, F.T. An Enhanced Minimax Loss Function Technique in Generative Adversarial Network for Ransomware Behavior Prediction. *Future Internet* **2023**, *15*, 318. https://doi.org/10.3390/fi15100318

Academic Editors: Weizhi Meng and Christian D. Jensen

Received: 29 July 2023
Revised: 7 September 2023
Accepted: 15 September 2023
Published: 22 September 2023

1. Introduction

Like other cyberattacks, ransomware attacks target a variety of systems and networks, including Personal Computers (PCs), mobile devices, Wireless Sensor Networks (WSN), Vehicular Ad Hoc Networks (VANETs), and the Internet of Things (IoT) [1,2]. Several studies have been conducted to detect ransomware attacks [3–5]. To detect crypto-ransomware early, the data collected during the pre-encryption phase of the crypto-ransomware lifecycle, before the encryption takes place is used [6,7]. The collected data are then used to train different machine learning algorithms to classify the programs into benign and ransomware [8]. However, the lack of sufficient data during the early phases of the attack adversely affects the accuracy of the model due to insufficient attack patterns [9].

Currently, ransomware attacks have targeted many Cyber Physical Systems (CPS), causing severe disruption of critical services and infrastructure [10,11]. In 2021, the US faced two significant CPS ransomware attacks on its critical infrastructure. The Colonial Pipeline, a major fuel supplier for the East Coast, experienced a cyberattack in May, leading to fuel shortages and panic buying in various states [12]. Then, in June, JBS, the world's top

meat supplier, was attacked, prompting plant shutdowns in the US and Australia. This attack utilized the Ryuk ransomware, demanding millions in ransom. Colonial Pipeline and JBS suffered significant financial losses, paying ransoms of $4.4 million and $11 million, respectively [12]. In October 2021, the Czech Republic's major power company, CEZ, was attacked with RansomExx ransomware after an intrusion via Winnti malware, causing power outages. Earlier, in December 2020, a natural gas facility was targeted using the TrickBot malware variant, prompting a response from the Cybersecurity and Infrastructure Security Agency (CISA) [10]. These attacks underline the severe consequences of ransomware on critical infrastructure, emphasizing the need for enhanced cybersecurity, and regular system updates, underscoring the significance of addressing vulnerabilities in CPS.

To detect ransomware, research can be grouped into three primary methods: process-centric, data-centric, and resource-centric [13]. The process-centric method observes running processes to spot suspicious patterns, utilizing machine learning classifiers like Random Forest and Naïve Bayes after gathering behavioral data. However, it often requires the complete runtime data for detection, making it less suitable for early detection. The data-centric method focuses on monitoring user data and files for abnormal changes, using techniques like decoy, entropy, and similarity measures [10]. Despite its intent, it cannot differentiate between benign program changes and crypto-ransomware actions, leading to high false alarm rates and compromising some user data before detection, rendering it ineffective for early crypto-ransomware identification.

The insufficient attack patterns are the main obstacle that degrades the early detection accuracy of ransomware attacks. Although several studies tried to overcome data insufficiency by focusing on how to select a subset of features that represent the immature ransomware attack patterns. Such approach assumes that the significance of those features remains unchanged. This does not hold as the polymorphic and metamorphic nature of the attack makes the ransomware generate different patterns every time it receives a new target. This is especially true during the pre-encryption stage where the goal of ransomware is to be evasive. Hence, the features become quickly obsolete. GAN has the potential to overcome the data insufficiency problem by augmenting the real attack patterns with artificial, yet realistic data. However, the Minimax loss function used by GANS's generator and discriminator is unable to estimate the distance between the probability distribution of real and artificial instances in the pre-encryption data of ransomware attacks. Hence, there is need for an improvement in the Minimax loss function, which is investigated by this study. The contribution of this paper is three-fold:

- We propose an enhanced GAN's generator loss function technique called Bi-Gradual Minimax, by incorporating a gradual up-weighting coefficient into the probability estimation calculation, which decreases the distance between the real and artificial distributions.
- We propose an improved GAN-based data augmentation module by incorporating the enhanced loss function in (1) into the GAN network, which generates artificial attack patterns that compensate for pre-encryption data insufficiency.
- We train an early detection model by training an LSTM estimator using the augmented dataset generated in (2), which improved the detection accuracy during the early pre-encryption phase of ransomware attacks.

The rest of this paper is organized as follows. Section 2 provides an overview of the related works. Section 3 details the methodology followed to design and develop the proposed techniques. Section 4 presents the experimental results, which are analyzed and discussed and compared with related works. The paper concludes with a summary of the methods and results as well as suggestions for future work, in Section 5.

2. Related Works

The major obstacle in the early detection of ransomware involves obtaining adequate data in the pre-encryption stage when the attack is still being set up and has not yet been executed [1,2]. Addressing this data shortage is crucial, as a sufficient dataset is needed

for precise early detection. Data augmentation is often used in machine learning solutions and presents a promising way to tackle this scarcity of data, a problem commonly faced by malware and ransomware early detection systems. To our understanding, no existing studies specifically address data augmentation in the pre-encryption stage of ransomware attacks [10]. Another challenge stems from the ever-changing nature of ransomware, which complicates the relevancy of features used for detection models [14]. For example, an attack pattern seen in one ransomware variant at a specific time might be more relevant than the same pattern displayed by a different variant at another time. This indicates that the importance of features can vary depending on the ransomware variant and the timing of the attack. Despite this, current early detection methods often operate on the assumption that the importance of these features remains constant, leading to "behavioral drift." This drift mostly results in detection systems becoming quickly outdated and less accurate over time.

The Generative Adversarial Network (GAN) has been widely used as an important component of deep neural networks [15]. The GAN model has gained massive attention from researchers recently due to its prominent characteristics. It has two main rewards for machine learning based models: the generality and adversarial [16]. It can generate new samples that can be used to prevent overfitting and, thus, improve machine learning performance. Moreover, it can be used to generate adversarial samples that can be used to improve the discriminability of the model. GAN use alternative training to estimate the density function over a data distribution using the Minimax algorithm [17]. The Minimax game algorithm tries to minimize the maximum possible loss which results in multiple possibilities that can be used to generate new samples. In doing so, GAN projects the available simple distribution to a much more complex high-dimensional, real-world data distribution [18]. GAN trains two adversarial networks called the generator and the discriminator. The generator is trained to map noise samples to synthetic samples with the goal is to generate new adversarial samples that can mislead the discriminator. Meanwhile, the discriminator trains to distinguish the real data samples from synthesized samples that were generated using the generator. GAN creates the new samples by making small changes to the original samples so as to deceive the detection model gain benefit of the nonlinear characteristics of neural networks and thus constructs a model that produces incorrect classification results.

Due to its prominent features, many researchers have applied the GAN algorithm to improve the classification performance of machine learning algorithms. Moti and Hashemi [15] proposed a malware detection model for Internet of Things (IoT) using the Generative Adversarial Network technique and Convolutional Neural Network (CNN). CNN was used to extract high-level features while GAN was used to generate new malware samples to mitigate the limitations of availability of insufficient malware samples in IoT. Li and Zhou [19] utilized GAN to develop a malware detection model-based adversarial example for the Android platform. Their proposed model called bi-objective GAN can generate evasive adversarial-example attacks able to fool the firewall and evade detection. Lu and Li [20] used GAN to improve the classification accuracy of the malware detection by generating new samples that can mimic realistic-like malware samples as well as the realistic distribution of data. Zhang and Zhou [17], proposed an improved Monte Carlo tree search (MCTS) algorithm for generating adversarial examples of cross-site scripting (XSS) attacks. A reward value is generated by the MCTS to rank the generated adversarial examples. The GAN algorithm was used to improve the detectability of adversarial examples. A GAN-based network was proposed to improve classification performance.

The following paragraph explains how GAN works. GAN formulates the adversarial problem as follows. Let X denote the sample space, x is a benign sample, and $g(x) > 0$ denotes the classification function when the result is benign. The attacker aims to generate a malware sample x^* that make $g(x^*) > 0$. Thus, the aim of the attacker can be formulated as follows:

$$x^* = \arg max_x \hat{g}(x), s.t. d\left(x, x^*\right) \leq d_{max}. \quad (1)$$

The GAN reduces the loss function value V during the training of both generator $\mathcal{G}$ and discriminator $\mathcal{D}$ by solving the following optimization function:

$$\min_{\mathcal{D}} \max_{\mathcal{G}} V(\mathcal{G}, \mathcal{D}) \tag{2}$$

where

$$V(\mathcal{G}, \mathcal{D}) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]. \tag{3}$$

The Z denotes the samples from noise distribution. Although the existing GAN has been effectively used to improve the performance of malware detection models, it does not fully fit for ransomware early detection due to data insufficiency that makes it difficult to perceive a clear probability distribution of the data. The unclear probability distribution prevents the GAN's generator from creating artificial samples as the discriminator will discard them due to the large distance between the probability distribution of artificial data and real data. GAN According to Dumoulin and Belghazi [21] and Uehara and Sato [22], existing GAN algorithms suffer from a vanishing gradient problem which leads to instability and model collapse due to the use of predefined adversarial loss function. Haloui and Gupta [23] used the derived approximation to the Wasserstein distance to improve the original GAN gradient-based loss function. The improved GAN algorithm is called WGAN. WGAN relies on the Arjovsky k-Lipschitz continuous function which adversely reduce the capacity of the discriminator model [24]. Gulrajani and Ahmed [25] anticipated an enhanced WGAN algorithm that penalizes the norm of discriminator gradients to train the discriminator network with respect to the sample data. There are several structure GAN algorithms including fully connected GANs [26], Conditional GANs [27], Convolutional GANs [28], GANs with inference models [27], and adversarial autoencoders [29]. Most of these algorithms use the standard loss function which suffers from the vanishing gradient problem and, thus, leads to instability and model collabs especially when insufficient data is used for training the classification task. Such limitations hinder the applications of the GAN algorithm to many challenging domains in cybersecurity such as early detection of ransomware attacks.

3. The Methodology

Figure 1 explains the architecture of our proposed BGM-GAN algorithm for ransomware detection. It consists of three main components: generator, discriminator, and loss functions. The generator learns artificial ransomware patterns based on feedback from the discriminator. It always tries to deceive the discriminator so as to identify the pre-encryption data as benign. Several components are needed for the generator to craft the artificial patterns, namely random inputs, generator network, discriminator network discriminator output, and generator loss. On the other hand, the discriminator is a classifier that distinguishes whether an instance of patterns is real or artificially created by the generator. The discriminator classification relies on training data coming from two different sources: real attack patterns and artificial patterns. The real patterns are used as positive instances whereas the artificial patterns are used as negative instances. The loss functions reflect the distance between the probability distributions of real and artificial patterns. In particular, there are two loss functions, one for the generator and the other for the discriminator. In this study, the Minmax function is used to control the upper limit of classification error by minimizing the maximum loss. This is suitable for the pre-encryption data that lack the sufficient attack patterns, where accurate estimation is challenging and classification error is inevitably high. In the following subsections, the design of the proposed algorithm, including the improved Minmax loss function will be elaborated. We start with a brief description of the GAN and CNN techniques.

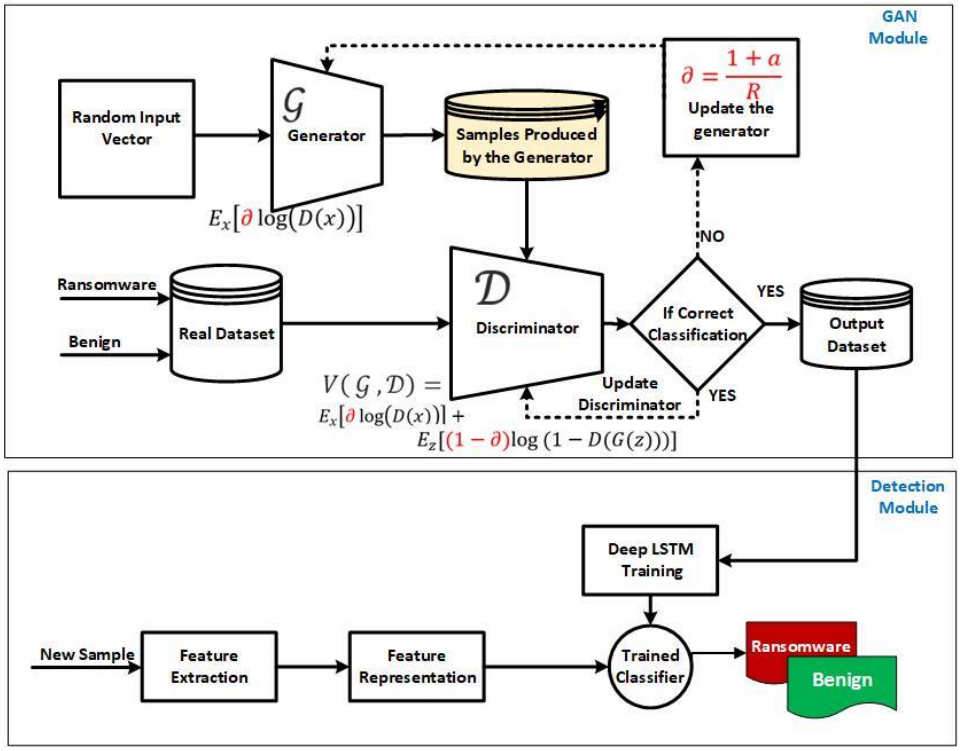


Figure 1. The design of Proposed BGM-GAN algorithm.

A. Generative Adversarial Networks:

Generative Adversarial Networks (GANs) have emerged as a groundbreaking development in the field of machine learning, particularly in the realm of deep learning. Initially conceptualized by Ian Goodfellow and his colleagues in 2014, GANs consist of two neural networks—a generator and a discriminator—that are trained simultaneously through a form of contest. While the generator strives to produce synthetic data that mimics a given data distribution, the discriminator aims to distinguish between genuine and synthetically generated data. This adversarial process leads to the continual refinement of both networks, enabling the generation of increasingly convincing synthetic data. The versatility of GANs has made them a subject of extensive research, with applications ranging from image and video generation to more complex tasks like data augmentation, anomaly detection, and even cybersecurity applications such as ransomware detection.

B. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have become a cornerstone in the field of machine learning, particularly for tasks related to image and video analysis. Introduced in the late 1980s and refined over subsequent decades, CNNs have demonstrated remarkable efficacy in the classification of visual data. Leveraging a specialized architecture that mimics the human visual cortex, CNNs are adept at automatically and adaptively learning spatial hierarchies of features from input images. Their ability to handle complex visual information has led to state-of-the-art results in a myriad of applications, from image and video recognition to medical image analysis. As a result, CNNs have become the go-to algorithm for problems involving such classifications.

3.1. The Improved Generative Adversarial Network Model for Ransomware Early Detection

The proposed algorithm is composed of two main phases: the GAN network and the early detection phase. The GAN network consists of two components, namely generator and discriminator. Figure 1 shows the design of the proposed algorithm. The following subsections explain the two phases.

3.1.1. Phase 1: The Enhanced Bi-Gradual Minimax Function of the Generative Adversarial Network

The purpose of GAN is to generate artificial pre-encryption attack patterns based on a dataset of real ransomware attacks. This will help in compensating for data insufficiency inherited in the data captured during the pre-encryption phase of ransomware attacks by augmenting the dataset with fake, yet realistic, samples.

When a sample is fed into the discriminator, it processes the information through its layers to make a prediction regarding the authenticity of the sample. Specifically, the discriminator assigns a probability score indicating how likely it believes the sample to be real rather than synthetic. During the training process, the generator continually improves its ability to create convincing synthetic data, while the discriminator fine-tunes its ability to distinguish between real and synthetic samples. This adversarial relationship between the two networks ensures that, over time, the generator becomes proficient at producing synthetic samples that are increasingly difficult to distinguish from real data, thus challenging the discriminator to continually improve its own performance.

Figure 2 shows the GAN architecture consisting of two components, the generator and the discriminator, that are trained together. The generator is responsible for creating artificial ransomware instances that resemble the real attacks. The new (artificial) ransomware instances are then sent to the discriminator that, in turns, tries to identify whether it is artificial. In this context, the generator tries to fool the discriminator by making the artificial sample as similar as the real ones as possible. Concretely, the generator creates the patterns with probability distribution similar to the probability distribution of the real ransomware.

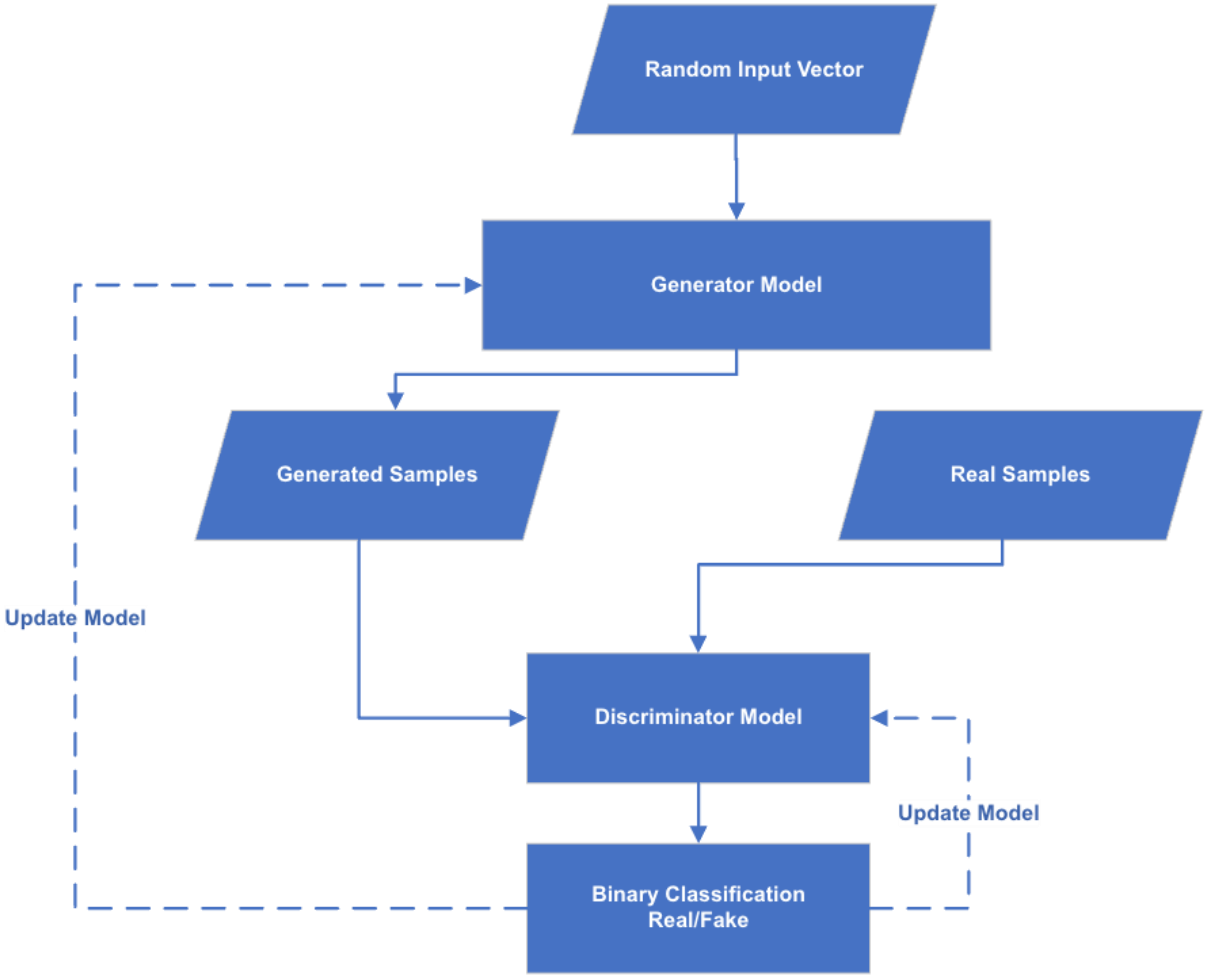


Figure 2. The general architecture of GAN.

The discriminator, on the other hand, is optimized to distinguish the real and artificial attack. If successfully identified, the feedback is sent to the generator, which will readjust its weights, that helps in creating new instances closer to the real ones. This process continues iteratively until both generator and discriminator are optimized. The process starts by feeding the generator with random input from an input vector. Based on this input the generator crafts an artificial attack pattern, which will then be sent to the discriminator. The discriminator will make the decision as to whether the data is ransomware or not. If the decision is correct, the generator is updated to improve its ability to generate more accurate attack patterns. Likewise, if the decision is wrong, the discriminator is updated to improve its ability to detect the artificial attack patterns. The process continues until both generator and discriminator are optimized as shown in the equation (4). At the end of this phase, a dataset that contains both real and artificial ransomware pre-encryption attacks instances are created. This dataset is used to train the ransomware early detection module in the second phase.

The Enhanced Bi-Gradual Minimax Loss Function

In the loss function schemes used by the generator and discriminator in the GAN algorithm, the distance between the probability distributions of real and artificial patterns are measured. Our proposed Bi-Gradual Minimax loss function improves the original Minimax function mentioned in Equation (1). Although the loss functions used by the generator and the discriminator rely on different probability distributions for error calculation, they are derived from the same formula as Equation (4).

$$V(\mathcal{G}, \mathcal{D}) = E_x[\partial \log(D(x))] + E_z[(1-\partial)\log(1-D(G(z)))] \quad (4)$$

where *D(x)* denotes the discriminator's estimate of the probability that real data instance x is real, *G(z)* denotes the generator's output given the noise z, *D(G(z))* denotes the discriminator's estimate of the probability that a fake instance is real, E_x represents the expected value over all real data instances, and E_z represents the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances *G(z))*. The ∂ denotes the gradual coefficient that controls the estimation of probability in both generator and discriminator. The gradual up-weighting coefficient is introduced by this study to cater for accurate loss estimation when the pre-encryption data are used to train a GAN-based early detection model and calculated by Equation (5) as follows:

$$\partial = \frac{1+a}{R} \quad (5)$$

where R is the number of real instances, while a is the number of artificial instances generated so far. As the number of real instances (R) is fixed, the value of coefficient ∂ changes proportionally to the number of artificial instances.

In the beginning when there are no or few artificial samples created, the value of ∂ starts small as the value of R in the denominator is way higher than the value of a in the numerator. When more artificial instances are generated, the value of ∂ grows. Therefore, the coefficient dynamically adjusts the upper limit of the error, so it starts low and increases progressively with the addition of new artificial instances. This helps the loss function to accurately estimate the probability that a particular instance is artificial even if the data is limited as in the case of ransomware early detection. The concept is that, in the early phase of ransomware attack, the distribution of attack patterns is not clear, so the discriminator tends to reject many samples that are not at enough close proximity to the real data. Therefore, the ∂ relaxes GAN discriminator's estimation criteria at the beginning when sufficient data representing the attack patterns are yet to be collected. When more artificial attack patterns are generated, the loss function becomes more stringent.

In the Minimax loss function, we incorporated the coefficient ∂ to dynamically adjust the upper limit of loss estimation. Therefore, the generator can adapt to the high variance

in the probability distribution of the pre-encryption phase of ransomware attacks. At the beginning of GAN's algorithm training, the number of artificial instances is low, which decreases the weight of the discriminator's term in the loss function in Equation (1). Therefore, the total loss will be calculated based on the generator's term. This gives the generator more freedom to craft attack patterns that can be accepted by the discriminator. The intuition is that when the influence of the discriminator's term in the loss function is low, the likelihood that it identifies the artificial patterns as real increases. In contrast, when the number of artificial instances of attack patterns increases, the influence of discriminator in Equation (1) increases, and the weight of the generator's term decreases. Hence, the ability of the discriminator to identify the artificial instances increases.

3.1.2. Phase 2: The Early Detection Module

In this phase, the augmented dataset generated during Phase 1 will be used to train the ransomware early detection model. The Long Short-Term Memory (LSTM) will be used to train a detection model. A set of informative features will be selected using the Mutual information Feature Selection (MIFS) according to Equation (6) as follows:

$$J(X_k) = I(X_k; Y) - \beta \sum_{X_j \epsilon S} I(X_j; X_k) + \gamma \sum_{X_j \epsilon S} I(X_j; X_k | Y) \quad (6)$$

where $I(X_k; Y)$ is the mutual information between the candidate feature X_k and the class label Y; $I(X_j; X_k | Y)$ is the conditional mutual information between the candidate feature X_k and the feature X_j in the selected set S, given the class label Y; while β and γ are parameters (coefficients) with values between 0 and 1. The purpose of feature selection is to reduce data dimensionality, which avoids the overfitting problem that negatively affects the detection accuracy. By selecting the most relevant features, the model will also generate less false alarms, which contributes to higher precision as well. Furthermore, reducing data dimensionality helps to decrease the complexity of the model, which makes it suitable for real-time scenarios like ransomware early detection. The MIFS will rank the attributes based on the entropy, such that those with higher entropy value get lower rank. Then, the MIFS will select n top-ranking features, that will be used as inputs in the LSTM algorithm.

The reason for using LSTM to build the detection module is due to its ability to remember the previous states, which is important property for ransomware early detection as it helps in tracking the progression of ransomware behavior during the pre-encryption phase. Therefore, the LSTM can trace the polymorphic behavior of the ransomware thanks to the cell state, which provides a bit memory to the LSTM so it can remember the past. The LSTM network is composed of three types of gates, namely input (Equation (7)), forget (Equation (8)), and output (Equation (9)):

$$i_t = \sigma(\omega_i[h_{t-1},\ x_t + b_i]), \quad (7)$$

$$i_f = \sigma(\omega_f[h_{t-1},\ x_t + b_f]), \quad (8)$$

$$i_o = \sigma(\omega_o[h_{t-1},\ x_t + b_o]) \quad (9)$$

where i_t, i_f, and i_o represent the input gate, forget gate, and the output gate. During the model's training, the LSTM will be trained using the data and selected features. The model is composed of several layers, namely input, output, and hidden. The number of nodes in the input layer is determined by the number of features selected by the MIFS. These nodes will receive data and process them to the hidden layer, after they multiply by the input weight. The hidden part of LSTM will be constructed using multiple layers. The number of hidden layers and nodes in each hidden layer will be optimized based on the bias factor during the training phase as well. In the hidden layers, the data will be processed based on the activation function used in the hidden nodes. The Relu function (Equation (10)) will be used as the activation function in all nodes in the hidden layers of LSTM, except the

layer that precedes the output, where the sigmoid function (Equation (11)) will be used. The output layer receives the data from the sigmoid functions in the last hidden layer and determine whether the instance is malicious or normal based on a threshold where values greater than 0.5 are considered attacks.

$$R(x) = \max(0, x) \tag{10}$$

$$\varnothing(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

The LSTM model is trained using the 10-fold cross validation method, where data is divided into two sets: training set and testing set. The training builds the model while testing evaluates the model's accuracy. The size of the training set is 90% of the data, and the testing set is 10% of the data. This process is repeated 10 times and the accuracy of the model will be recorded. At the end of training/testing process, the averaged accuracy is calculated, which determines the overall accuracy of the model.

4. Results and Discussion

In this section, the description of our development environment setup is provided where in the experiments were conducted as well as the dataset used for those experiments. Then, the results of each technique are discussed along with the comparison to related works.

4.1. Experimental Facilities

Our evaluation was conducted using several Python-based packages including Sklearn [Version Number: 1.3.1], Pandas [Version Number: 2.1.0], Numpy [Version Number: 1.25.0], and SkFeature [Version Number: 1.0.0]. TensorFlow [Version Number: 2.3] was used to build both the generator and discriminator networks of the GAN model. Anaconda software [Version Number: 3] was used as a Python IDE. Specialized libraries like Keras simplified the architecture development process, allowing us to focus on optimizing the model rather than dealing with lower-level details. For data preprocessing and feature extraction, tools such as scikit-learn, Numpy, and Pandas were utilized.

4.2. Corpus of Crypto-Ransomware Binaries

The corpus of crypto-ransomware binaries used in this study were downloaded from http://www.virusshare.com (accessed on 14 March 2023) public repository [30–33]. The corpus consists of 8152 samples. Those samples represent different families such as Cerber, TeslaCrypt, CryptoWall, Petya, and WannaCry. The samples were collected during the period from September 2016 to August 2017. In addition, 1000 benign programs were downloaded from informer.com [32–35], a popular Windows-based applications repository. Those files are called ransomware binaries, often referred to as Portable Executable (PE) files. Those PE files are a central focus in the field of malware analysis as they serve as a common format for executables, object code, and DLLs in Windows operating systems. When analyzing ransomware, PE files are closely inspected statically or dynamically to understand the malicious software's structure, functionalities, behavior, and potential impact. By dissecting the PE file, we can extract valuable metadata, identify suspicious or known malicious signatures, and gain insight into the ransomware's behavior, such as encryption methods, communication with command and control servers, or persistence mechanisms. Analyzing these binaries is pivotal in developing countermeasures, creating detection signatures, and understanding the evolving tactics of ransomware authors. In this study, the dynamic analysis, which involves executing the PE file as the sandbox, is adopted.

4.3. Data Engineering

The benign application group consists of file utilities like compression, encryption, and editing tools, along with office, developer, multimedia tools, drivers, browsers, and games. Each sample was verified using VirusTotal, which employs 56 anti-virus (AV) programs to ascertain each sample's maliciousness, as referenced in multiple studies. Samples identified as malicious by less than 5 AV engines were disregarded. The benign samples were only selected if all 56 AV engines marked them as trusted. Both ransomware and benign software were run and studied in real-time within the Cuckoo Sandbox. Once a sample was introduced into the analysis machine, the sandbox monitored the resulting process, recording the APIs into a dedicated trace file for that sample. The information from these files formed the main body of data.

4.4. Feature Selection

The Enhanced Mutual Information Features Selection (EMIFS) built in our previous work was used to select the most informative features for the pre-encryption phase of crypto-ransomware lifecycle. The experiments were conducted using several epochs sets with different number of features, i.e., 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50 features. The results show the accuracy at different numbers of epochs. The dataset was divided into a training set and testing set using a 10-fold cross validation approach. The testing set was then used to determine the classification accuracy of those classifiers.

4.5. Compilation and Comparison of Results

Table 1 shows the accuracy results based on number of epochs. It can be observed that for the number of epochs less than 90, the classification accuracy increases when the size of features set increases from 5 to 20 features. However, when the size of features increase, the accuracy drops again. When the number of epochs increases to 120, the accuracy increases until the number of features reaches 25. After that, no significant increase was shown. If the number of epochs increased to 150, 180, and 210, the accuracy increases for the feature sets 10, 15, 20, and 35. When the number of features increases to 35 and above, the accuracy does not improve. Similarly, the accuracy when the feature set size is 5 fluctuates between 0.88 and 0.919.

Table 1. Accuracy results of the proposed BGM-GAN based on number of epochs.

Features \ Epochs	E30	E60	E90	E120	E150	E180	E210
5	0.890	0.906	0.934	0.932	0.942	0.938	0.926
10	0.908	0.920	0.951	0.935	0.954	0.953	0.957
15	0.898	0.939	0.937	0.944	0.950	0.969	0.961
20	0.915	0.959	0.956	0.954	0.956	0.965	0.962
25	0.910	0.949	0.941	0.973	0.971	0.979	0.969
30	0.927	0.960	0.931	0.963	0.980	0.967	0.967
35	0.936	0.953	0.938	0.970	0.979	0.970	0.981
40	0.937	0.925	0.953	0.971	0.976	0.991	0.986
45	0.939	0.916	0.946	0.984	0.981	0.991	0.992
50	0.938	0.933	0.922	0.989	0.984	0.994	0.995

To show the efficacy of our proposed BGM-GAN techniques, the results were compared with the GAN-based detection model proposed by [36]. Furthermore, the comparison was conducted using the CNN and DBN models proposed by [37,38]. Moreover, the experiments were conducted using different features set sizes ranging from 5 to 50 and incremented by 5 features between each of the two consequent sets. The classification accuracy was used as the measurement of the classification performance. Figure 3 shows the comparison results between our proposed BGM-GAN and the related techniques. Based on the comparison results, the proposed BGM-GAN outperforms GAN, CNN, and DBN.

Figure 3a shows an accuracy comparison of our proposed BGM-GAN with the related models (standard GAN, CNN, and DBN) across various feature counts, with each model trained for 30 epochs. Our proposed GAN demonstrates a general increase in accuracy as the feature count rises, peaking at 45 features with an accuracy of 0.939. There is a slight dip at 50 features (0.938), but the proposed GAN still displays consistent high performance. The standard GAN begins at a similar accuracy to our proposed GAN (0.891 at 5 features), with its highest accuracy achieved at 25 features (0.930). Post this peak, the accuracy diminishes slightly, reaching accuracy of 0.916 at 50 features. The CNN model starts with an accuracy of 0.865 at 5 features and reaches its peak accuracy at 30 features (0.929). Past this point, the model's accuracy regresses slightly, ending with an accuracy of 0.919 at 50 features. The DBN model, meanwhile, shows less fluctuation. It starts with 0.885 accuracy at 5 features and peaks at 20 and 30 features with an accuracy of 0.914. Thereafter, it maintains a relatively stable performance, ending with 0.906 at 50 features. In summary, all models exhibit peaks in performance at different feature counts. Our proposed BGM-GAN, however, consistently outperforms the others across the range of features, reaching the highest accuracy of all models at 45 features (0.939), as shown in Figure 3.

Figure 3b shows the comparison of the accuracy of our proposed BGM-GAN and the related models, standard GAN, CNN, and DBN, when trained for 60 epochs across a varying number of features. The proposed GAN shows an increasing trend in accuracy as the number of features goes up to 30, peaking at an impressive 0.960 accuracy. Beyond 30 features, however, its accuracy slightly decreases, with the lowest recorded at 45 features (0.916), then showing an increase at 50 features (0.933). The standard GAN model shows fluctuating performance, starting with 0.895 at 5 features and reaching its highest accuracy at 45 features (0.940). The CNN model exhibits a somewhat inconsistent performance, beginning with an accuracy of 0.872 at 5 features, peaking at 25 features (0.915), then experiencing a slight decrease with further increase in features, achieving an accuracy of 0.925 at 50 features. The DBN model starts with 0.894 accuracy at 5 features, peaking twice at 25 and 35 features with an accuracy of 0.928 and 0.930, respectively. It then shows a dip at 40 features, and a slight increase to reach 0.922 at 50 features. All models show variations in performance at different feature counts. Nevertheless, the proposed GAN consistently outperforms the other models in most feature counts, achieving the highest accuracy of all models at 30 features (0.960).

Figure 3c illustrates the accuracy comparison between our proposed BGM-GAN and the related works (standard GAN, CNN, and DBN) across varying numbers of features. All models were trained for 90 epochs. The proposed GAN exhibits high accuracy starting from 0.934 at 5 features, peaking at 10 features with an accuracy of 0.951. While there is a slight decrease in accuracy at 30 features (0.931), it picks up again reaching a second peak at 40 features (0.953) and then decreases slightly towards 50 features (0.922). The standard GAN begins with 0.882 accuracy at 5 features and peaks at 30 features (0.927). Its performance remains fairly stable thereafter, with the final accuracy at 50 features being 0.927. The CNN model's performance starts at 0.874 accuracy for 5 features and progressively increases to reach a peak at 40 features (0.924), and then slightly decreases at 50 features (0.916). The DBN model exhibits a steady increase in performance, from an initial 0.884 accuracy at 5 features to its peak at 30 features (0.932). Thereafter, the accuracy decreases, reaching 0.916 at 50 features. All the models exhibit fluctuations in accuracy at different feature counts. However, the proposed GAN consistently shows higher performance in comparison to the other models in most feature counts, reaching the highest accuracy of all models at 10 and 40 features (0.951 and 0.953, respectively).

Figure 3d displays an accuracy comparison between our proposed BGM-GAN and the related works (standard GAN, CNN, and DBN), across varying numbers of features. All models were trained for 120 epochs. Starting with the proposed GAN, it shows a consistent increase in accuracy with a slight dip at 30 features (0.963), before reaching the highest accuracy of 0.989 at 50 features. The standard GAN shows fluctuations with its performance peaking at 35 features with an accuracy of 0.940. However, it ends with a slightly lower

accuracy of 0.927 at 50 features. The CNN model displays an initial increase in accuracy, peaking at 25 features (0.940). After this peak, there is a decrease at 30 features (0.915) followed by a minor increase and a subsequent drop, ending with the accuracy of 0.899 at 50 features. Finally, the DBN model starts at 0.908 at 5 features shows its peak performance at 35 features (0.946) and then displays a slight decrease, ending with an accuracy of 0.929 at 50 features. The proposed GAN outperforms the other models across the range of features, achieving the highest accuracy of all models at 50 features (0.989). The standard GAN, CNN, and DBN models demonstrate fluctuating performance across different feature counts, with their peak accuracies achieved at 35, 25, and 35 features, respectively.

Figure 3e shows an accuracy comparison between our proposed BGM-GAN and the related models (standard GAN, CNN, and DBN), across varying numbers of features. All these models were trained for 150 epochs. The proposed GAN displays a consistent increase in accuracy, starting from 0.942 at 5 features and peaking at 0.984 at 50 features. The accuracy dips slightly at 15 features (0.950), then increases consistently, except for a minor decrease at 40 features (0.976). The standard GAN shows varying performance, with its highest accuracy achieved at 25 features (0.961) and 50 features (0.945), and the lowest accuracy at 15 features (0.879). The CNN model exhibits a steady increase in accuracy, peaking at 45 features (0.949), followed by a substantial decrease to 0.899 at 50 features. The DBN model displays a somewhat fluctuating performance, beginning at 0.882 for 5 features, peaking at 35 features (0.944), and ending with an accuracy of 0.930 at 50 features. It can be observed that the proposed GAN consistently outperforms the other models across the range of features, achieving the highest accuracy of all models at 50 features (0.984). The standard GAN, CNN, and DBN models demonstrate variable performance across different feature counts, with their peak accuracies occurring at 25, 45, and 35 features, respectively.

Figure 3f shows the accuracy comparison between our proposed BGM-GAN and the related works (standard GAN, CNN, and DBN) over various feature counts. Each model was trained for 180 epochs. The proposed GAN shows consistently high accuracy. Starting at 0.938 at 5 features, the accuracy generally increases to reach the highest accuracy of 0.994 at 50 features. There is a slight dip at 30 features (0.967), but the model then continues to increase its performance. The standard GAN has some fluctuation, starting with an accuracy of 0.887 at 5 features, reaching a peak at 20 features (0.939), and ending at 0.930 for 50 features. The CNN model also fluctuates, with its highest accuracy achieved at 30 features (0.952) and the lowest at 5 features (0.882). Its accuracy dips after 30 features and ends at 0.935 for 50 features. The DBN model starts at 0.892 at 5 features, peaks at 30 features (0.947), then slightly decreases, ending with an accuracy of 0.931 at 50 features. It can be noticed that the proposed GAN consistently outperforms the other models across the range of features, reaching the highest accuracy at 50 features (0.994). The standard GAN, CNN, and DBN models display more fluctuation in their performance across different feature counts, with the peak accuracies achieved at 20, 30, and 30 features, respectively.

Figure 3g depicts an accuracy comparison between our proposed BGM-GAN and the related works (standard GAN, CNN, and DBN) for different numbers of features. All models were trained for 210 epochs. The proposed GAN shows a consistent increase in accuracy as the number of features increases, beginning at 0.926 at 5 features and reaching a peak at 0.995 at 50 features, indicating the robustness and efficacy of our proposed model. The standard GAN, on the other hand, exhibits a fluctuating performance with its highest accuracy achieved at 30 features (0.979). The accuracy slightly decreases afterwards, but it still manages to end with a relatively high accuracy of 0.977 at 50 features. The CNN model starts at 0.905 at 5 features and peaks at 25 features with an accuracy of 0.959. Thereafter, the accuracy tends to decrease with a minor increase at 40 features (0.967), finally ending with an accuracy of 0.940 at 50 features. The DBN model starts at 0.927 at 5 features, dips slightly at 10 features (0.897), and then rises steadily till 25 features (0.950). It further increases to 0.940 at 30 features, then experiences fluctuations, eventually ending with an accuracy of 0.935 at 50 features. It can be observed that the proposed GAN outperforms the other models across the feature range, reaching the highest accuracy of all

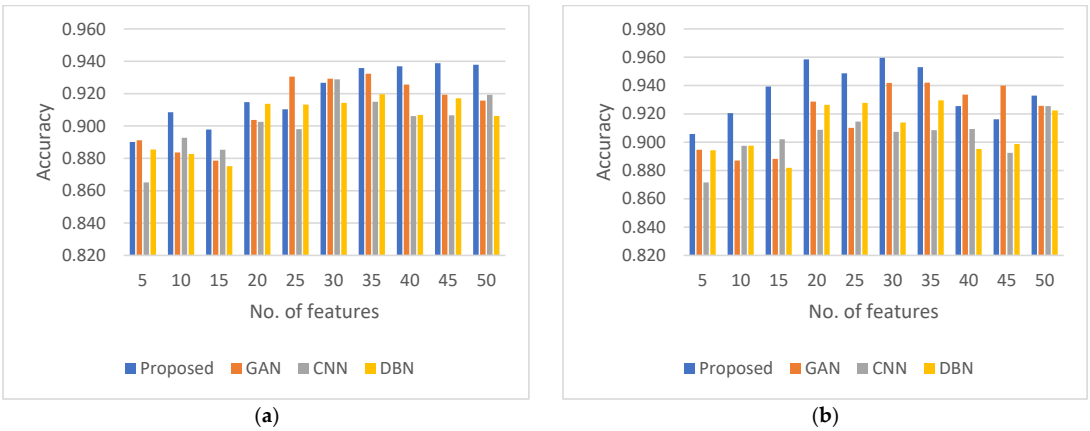
models at 50 features (0.995). The standard GAN, CNN, and DBN models display variable performance across different feature counts, with their peak accuracies achieved at 30, 25, and 25 features, respectively.

Figure 3a–g shows that our proposed BGM-GAN technique achieved detection accuracy higher than the GAN-based model of [36] for all epochs. The proposed technique also outperformed the accuracy of other deep learning models like CNN and DBN. This is attributed to the ability of the proposed Bi-Gradual Minimax loss function in estimating the distance between the probability distributions of real and artificial instances in the pre-encryption data of ransomware attacks where the data that represent ransomware's early behavior is insufficient. This supports our assumption that the existing works failed to consider the effect of data insufficiency on the generator and discriminator, which makes them unable to produce artificially realistic metamorphic ransomware samples.

The comparison results shown in Figure 3a–g also show that the accuracy increases by increasing the number of features up to a certain limit. When the limit is reached, the accuracy either drops or show no significant improvement. The limit to which the number of selected features contributes to improving the accuracy varies based on the number of epochs in each run. For example, when the model is trained using 30 epochs, the maximum accuracy was achieved when the number of the selected features was 45, while for 60 epochs the maximum accuracy was achieved using 30 features.

The reason that the accuracy drops when number is the overfitting caused by increased number of features and number of epochs during model training. When the number of features increases, the data dimensionality increases as well. Such data dimensionality negatively affects the accuracy of the detection model. The issue is aggravated when dealing with data that have insufficient ransomware behavior during the early phases. Likewise, when the number of epochs during the model's training increases, the likelihood that overfitting occurs also increases which makes the loss function unable to accurately measure the distance between real and artificial patterns. This explains the deterioration of the accuracy after a certain number of features and epochs.

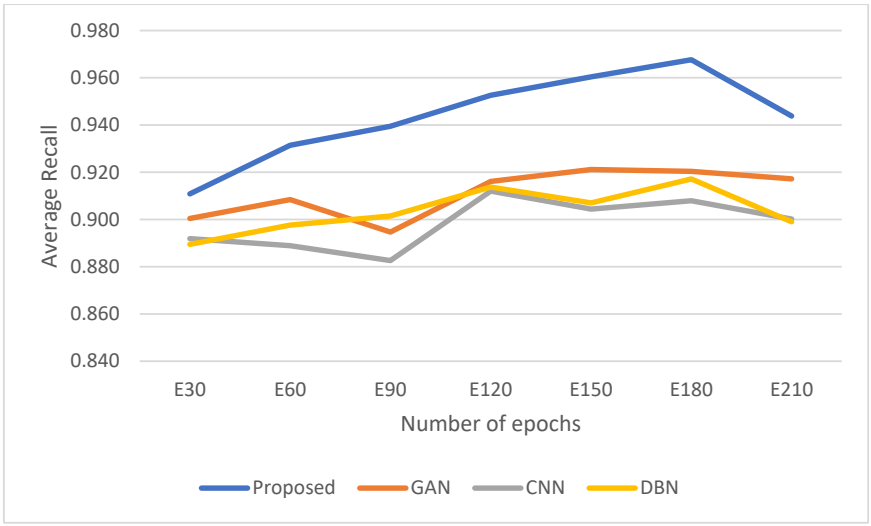
Likewise, Figure 4a,b show that on average, our proposed BGM-GAN outperformed the existing techniques in terms of recall and false positive rate, respectively. The comparison was carried out by measuring both the recalls and the false positive rate over several number of epochs using several feature set sizes (as has been done in Figure 3). The average was taken with respect to the feature set size. The improvement shown by BGM-GAN is attributed to the ability of the BGM method to capture the early patterns even when less data are available. This conforms with the results shown in Figure 3. Additionally, Table 2 shows the training time that the BGM-GAN takes per number of epochs. We observed that the training time was between 2 to 4 milliseconds (ms) per step and around 2 s per epoch. Therefore, the greater number of epochs the training uses, the more time it takes to develop the model.



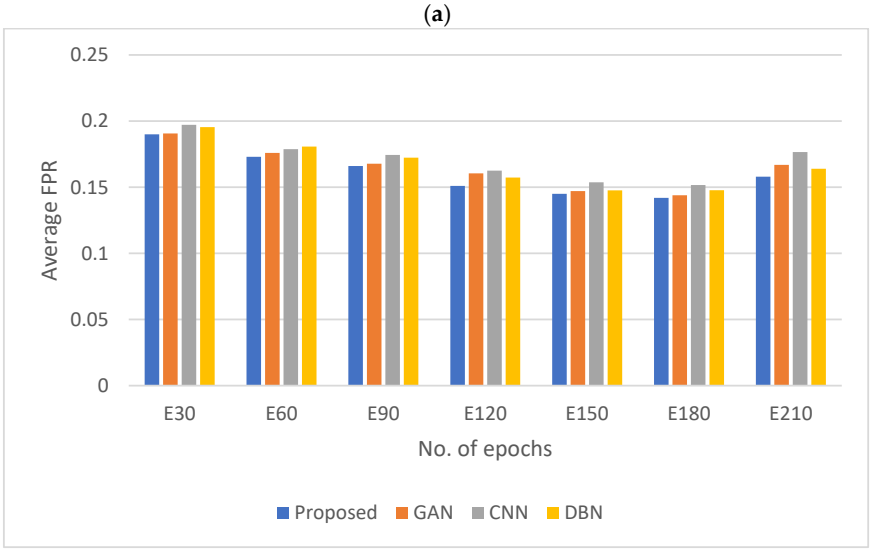
(**a**) (**b**)

Figure 3. *Cont.*

Figure 3. (**a**) Comparison of accuracy between our proposed BGM-GAN and the related models, standard GAN, CNN, and DBN, when trained for 30 epochs across a varying number of features. (**b**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 60 epochs across a varying number of features. (**c**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 90 epochs across a varying number of features. (**d**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 120 epochs across a varying number of features. (**e**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 150 epochs across a varying number of features. (**f**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 180 epochs across a varying number of features. (**g**) Comparison of accuracy between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 210 epochs across a varying number of features.



(**a**)



(**b**)

Figure 4. (**a**) Comparison of the averaged recall (true positive rate) between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, w across a varying number of epochs. (**b**) Comparison of the averaged false positive rate (FPR) between our proposed BGM-GAN and related models, standard GAN, CNN, and DBN, when trained for 60 epochs across a varying number of epochs.

Table 2. Training time for various number of epochs.

# of Epochs	Total Training Time (Seconds)
30	70 s
60	132 s
90	189 s
120	253 s
150	307 s
180	369 s
210	431 s

To sum up, the novelty of this work lies in addressing the critical challenge of early detection of ransomware attacks, which is often compromised due to insufficient and

quickly outdated attack patterns. Unlike previous studies that focused on selecting a subset of features that quickly become obsolete due to the dynamic nature of ransomware, this study explores a fundamentally different approach using GAN. The paper introduces an enhanced loss function for the GAN's generator by incorporating a gradual up-weighting coefficient, aimed at minimizing the distance between the real and artificially generated data distributions. This technique helps our GAN model to create synthetic ransomware samples that are close to the real attack patterns, overcoming the lack of real attack patterns during the pre-encryption phase. This enhanced loss function is incorporated into a GAN-based data augmentation module for generating more representative attack patterns. Notably this technique improves our measurements across the board. Our model contributes to a more robust and timely ransomware detection outcome, filling a gap in existing cybersecurity literature and best practice.

5. Conclusions

In this study, we have introduced the Bi-Gradual Minimax (BGM) loss function, a novel optimization strategy designed to enhance the performance of Generative Adversarial Networks (GANs) in the context of ransomware early detection. By integrating this advanced loss function into the GAN framework, we have achieved dual objectives: a more nuanced minimization of maximum loss for both the generator and the discriminator. This advancement serves a twofold purpose. First, it enables the generator to craft highly convincing synthetic patterns that closely mimic the pre-encryption data distribution, thereby addressing the limitations imposed by data scarcity. Second, it empowers the discriminator to identify potentially malicious activities more accurately, based on these refined synthetic patterns. One of the most compelling aspects of the BGM-GAN model is its synergistic architecture. Improvements in either the generator or the discriminator contribute reciprocally to the enhanced accuracy of the overall system as shown here in our results. Furthermore, the gradual up-weighting approach employed by our proposed method ensures that our BGM-GAN model is particularly effective when data is scarce, a frequent challenge in the domain of ransomware early detection. Therefore, the BGM-GAN model not only addresses the issue of data insufficiency but also provides a robust framework for predicting future ransomware attack patterns with higher accuracy.

Author Contributions: All authors have contributed equally to the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Najran University and the authors would like to thank them for their support.

Data Availability Statement: All the ransomware PE files are publicly available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Urooj, U.; Maarof, M.A.B.; Al-rimy, B.A.S. A proposed adaptive pre-encryption crypto-ransomware early detection model. In Proceedings of the 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021.
2. Ahmed, Y.A.; Huda, S.; Al-Rimy, B.A.S.; Alharbi, N.; Saeed, F.; Ghaleb, F.A.; Ali, I.M. A weighted minimum redundancy maximum relevance technique for ransomware early detection in industrial IoT. *Sustainability* **2022**, *14*, 1231. [CrossRef]
3. Assaggaf, A.M.A.; Al-Rimy, B.A.; Ismail, N.L.; Al-Nahari, A. Development of Graph-Based Knowledge on Ransomware Attacks Using Twitter Data. In Proceedings of the International Conference on Data Science and Emerging Technologies, Virtual, 20–21 December 2022; Springer: Berlin/Heidelberg, Germany.
4. Olaimat, M.N.; Maarof, M.A.; Al-rimy, B.A.S. Ransomware anti-analysis and evasion techniques: A survey and research directions. In Proceedings of the 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021.
5. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-Rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *J. Netw. Comput. Appl.* **2020**, *167*, 102753. [CrossRef]
6. Al-rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Gener. Comput. Syst.* **2019**, *101*, 476–491. [CrossRef]

7. Al-Rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Alsolami, F.; Shaid SZ, M.; Ghaleb, F.A.; Al-Hadhrami, T.; Ali, A.M. A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. *IEEE Access* **2020**, *8*, 140586–140598. [CrossRef]
8. Al-rimy, B.A.S.; Maarof, M.A.; Prasetyo, Y.A.; Shaid SZ, M.; Ariffin AF, M. Zero-day aware decision fusion-based model for crypto-ransomware early detection. *Int. J. Integr. Eng.* **2018**, *10*. [CrossRef]
9. Al-rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. A 0-day aware crypto-ransomware early behavioral detection framework. In *Recent Trends in Information and Communication Technology: Proceedings of the 2nd International Conference of Reliable Information and Communication Technology (IRICT 2017), Johor Bahru, Malaysia, 23–24 April 2017*; Springer: Berlin/Heidelberg, Germany, 2018.
10. Gazzan, M.; Sheldon, F.T. Opportunities for Early Detection and Prediction of Ransomware Attacks against Industrial Control Systems. *Future Internet* **2023**, *15*, 144. [CrossRef]
11. Gazzan, M.; Alqahtani, A.; Sheldon, F.T. Key Factors Influencing the Rise of Current Ransomware Attacks on Industrial Control Systems. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021.
12. Alqahtani, A.; Sheldon, F.T. A survey of crypto ransomware attack detection methodologies: An evolving outlook. *Sensors* **2022**, *22*, 1837. [CrossRef]
13. Urooj, U.; Al-Rimy, B.A.S.; Zainal, A.; Ghaleb, F.A.; Rassam, M.A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Appl. Sci.* **2022**, *12*, 172. [CrossRef]
14. Aboaoja, F.A.; Zainal, A.; Ghaleb, F.A.; Al-rimy, B.A.S. Toward an ensemble behavioral-based early evasive malware detection framework. In Proceedings of the 2021 International Conference on Data Science and Its Applications (ICoDSA), Virtual, 10–11 April 2021.
15. Moti, Z.; Hashemi, S.; Karimipour, H.; Dehghantanha, A.; Jahromi, A.N.; Abdi, L.; Alavi, F. Generative adversarial network to detect unseen internet of things malware. *Ad. Hoc. Netw.* **2021**, *122*, 102591. [CrossRef]
16. Yinka-Banjo, C.; Ugot, O.-A. A review of generative adversarial networks and its application in cybersecurity. *Artif. Intell. Rev.* **2020**, *53*, 1721–1736. [CrossRef]
17. Zhang, X.; Zhou, Y.; Pei, S.; Zhuge, J.; Chen, J. Adversarial examples detection for XSS attacks based on generative adversarial networks. *IEEE Access* **2020**, *8*, 10989–10996. [CrossRef]
18. Wang, C.; Xu, C.; Yao, X.; Tao, D. Evolutionary generative adversarial networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 921–934. [CrossRef]
19. Li, H.; Zhou, S.; Yuan, W.; Li, J.; Leung, H. Adversarial-example attacks toward android malware detection system. *IEEE Syst. J.* **2019**, *14*, 653–656. [CrossRef]
20. Lu, Y.; Li, J. Generative adversarial network for improving deep learning based malware classification. In Proceedings of the 2019 Winter Simulation Conference (WSC), National Harbor, MD, USA, 8–11 December 2019.
21. Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; Courville, A. Adversarially learned inference. *arXiv* **2016**, arXiv:1606.00704.
22. Uehara, M.; Sato, I.; Suzuki, M.; Nakayama, K.; Matsuo, Y. Generative adversarial nets from a density ratio estimation perspective. *arXiv* **2016**, arXiv:1610.02920.
23. Haloui, I.; Gupta, J.S.; Feuillard, V. Anomaly detection with Wasserstein GAN. *arXiv* **2018**, arXiv:1812.02463.
24. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014.
25. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A.C. Improved training of wasserstein gans. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
26. Barua, S.; Erfani, S.M.; Bailey, J. FCC-GAN: A fully connected and convolutional net architecture for GANs. *arXiv* **2019**, arXiv:1905.02417.
27. Li, M.; Lin, J.; Meng, C.; Ermon, S.; Han, S.; Zhu, J.Y. Efficient spatially sparse inference for conditional gans and diffusion models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 28858–28873. [CrossRef]
28. Torfi, A.; Fox, E.A.; Reddy, C.K. Differentially private synthetic medical data generation using convolutional GANs. *Inf. Sci.* **2022**, *586*, 485–500. [CrossRef]
29. Hoang, T.-N.; Kim, D. Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders. *Veh. Commun.* **2022**, *38*, 100520. [CrossRef]
30. Le Guernic, C.; Legay, A. Ransomware and the Legacy Crypto API. In Proceedings of the Risks and Security of Internet and Systems: 11th International Conference (CRiSIS 2016), Roscoff, France, 5–7 September 2016; Revised Selected Papers. Springer: Berlin/Heidelberg, Germany, 2017.
31. Christensen, J.B.; Beuschau, N. Ransomware Detection and Mitigation Tool. Master's Thesis, Technical University of Denmark, Kongens Lyngby, Denmark, 2017.
32. Chen, Z.-G.; Kang, H.S.; Yin, S.N.; Kim, S.R. Automatic Ransomware Detection and Analysis Based on Dynamic API Calls Flow Graph. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems, Krakow, Poland, 20–23 September 2017; ACM: Krakow, Poland, 2017; pp. 196–201.

33. Sgandurra, D.; Muñoz-González, L.; Mohsen, R.; Lupu, E.C. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *arXiv* **2016**, arXiv:1609.03020.
34. Ioanid, A.; Scarlat, C.; Militaru, G. The Effect of Cybercrime on Romanian SMEs in the Context of Wannacry Ransomware Attacks. In Proceedings of the 12th European Conference on Innovation and Entrepreneurship ECIE 2017, Paris, France, 21–22 September 2017.
35. Pandey, S.K.; Mehtre, B.M. Performance of malware detection tools: A comparison. In Proceedings of the 2014 IEEE International Conference on Advanced Communication, Control and Computing Technologies, ICACCCT 2014, Ramanathapuram, India, 8–10 May 2014; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2015.
36. Zhang, X.; Wang, J.; Zhu, S. Dual Generative Adversarial Networks Based Unknown Encryption Ransomware Attack Detection. *IEEE Access* **2022**, *10*, 900–913. [CrossRef]
37. Yadav, P.; Menon, N.; Ravi, V.; Vishvanathan, S.; Pham, T.D. EfficientNet convolutional neural networks-based Android malware detection. *Comput. Secur.* **2022**, *115*, 102622. [CrossRef]
38. Su, X.; Shi, W.; Qu, X.; Zheng, Y.; Liu, X. DroidDeep: Using Deep Belief Network to characterize and detect android malware. *Soft Comput.* **2020**, *24*, 6017–6030. [CrossRef]